INSA ALBERT EINSTEIN PROFESSORSHIP

PUBLICATIONS

G.N. Ramachandran

Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

VOLUME - X MR 65 - 68

CONTENTS

MATLOG Program in FORTRAN

Part IV : Refinement of classical logic treated algorithmically in BVMF and computer tested using MATLOG subroutines

G.N. Ramachandran
INSA Albert Einstein Professor

and

T.A. Thasneem
Visiting Fellow

MATLOG Program in FORTRAN

Part IV : Refinement of classical logic treated algorithmically in BVMF and computer tested using MATLOG subroutines

G.N. Ramachandran
INSA Albert Einstein Professor

and

T.A. Thanaraj*
Visiting Fellow

Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

*(Permanent Address: Centre for Cellular & Molecular Biology, Hyderabad 500 007.)

Matphil Reports No. 65, November 1987

CONTENTS

MATLOG Program in FORTRAN

Part IV: Refinement of classical logic treated algorithmically in BVMF and computer tested using MATLOG subroutines

1. Introduction

This report effectively fills in the gaps in the theoretical analysis of various tests carried out in Parts I, II and III. They essentially deal with two aspects.

(a) Test of conjunction identity, which, in the case of SNS, is equivalent to finding the conditions on $\underline{\underline{P}}$, $\underline{\underline{Q}}$, $\underline{\underline{R}}$ in Eqs. (1) and (2) below in which Eq.(1) is satisfied, or definitely not satisfied. This is discussed for SNS in MR-60 under Problem 4 and for QL-2 in MR-61 under Problem 8. A more general case of GLOGIC is discussed in MR-62 and some general results are given therein. The most important of these is the fact that the l.h.s and the r.h.s of Eq.(3a) need not be the same, in general, for GLOGIC.

$$(\underline{\underline{a}}\ \underline{\underline{P}}\ \underline{\underline{b}}) \wedge (\underline{\underline{a}}\ \underline{\underline{Q}}\ \underline{\underline{b}}) = (\underline{\underline{a}}\ \underline{\underline{R}}\ \underline{\underline{b}}) \tag{1}$$

where

$$\underline{\underline{P}} \otimes \underline{\underline{Q}} = \underline{\underline{R}} \tag{2}$$

$$(\underline{\underline{\underline{a}}}\ \underline{\underline{\underline{P}}}\ \underline{\underline{\underline{b}}}) \wedge (\underline{\underline{\underline{a}}}\ \underline{\underline{\underline{Q}}}\ \underline{\underline{\underline{b}}}) = (\underline{\underline{\underline{a}}}\ \underline{\underline{\underline{R}}}\ \underline{\underline{\underline{b}}}) \tag{3a}$$

where

$$\underline{\underline{\underline{P}}} \oplus \underline{\underline{\underline{Q}}} = \underline{\underline{\underline{R}}} \tag{3b}$$

Although, in general, the identity of the two sides of Eq.(1) or Eq.(3a) is not required, it is equally interesting to examine what are the conditions under which they are in fact equal, and this was discussed partly in MR-62 and will be discussed in detail in Sections 2 and 3 below. Interestingly, the nature of the differences between the l.h.s and r.h.s and the pinpointing of the precise origin of these differences is also an interesting problem. This is discussed in Section 4. In all these sections, the theoretical analysis is supported by detailed computational outputs in SNS. We believe that the treatment given here is a good illustration of how very subtle differences between combinations of logical relations can be incorporated in computer programs via BVMF and MATLOG.

Section 6 will deal also with a very subtle point, namely the equivalence, or otherwise, of truth values calculated via pure BVMF and via the BVMF implementation of classical logical techniques. This is illustrated best by the two MATLOG functions SSTV and SSTV2 in SNS, and correspondingly QSTV and QSTV2 in QL-2. These have been briefly dealt with in Part I

and Part II, but the treatment can be made more general using GLOGIC and using Boolean truth values, rather than SNS truth values, as in the functions GBTV and GBTV2. This is done in Section 6, and it turns out that the differences in all cases arise only for combinations of a permissible state with the non-permissible contradictory state (X in SNS and XXX in QL-2) being used as the input for the truth value calculation. Although this will never occur in a practical problem, the corresponding case of a null set in set theory is quite relevant in GLOGIC and is therefore to be considered. Further, in order to make the BVMF theory foolproof and universally consistent, such differences should be investigated and their precise origins pointed out, so that the formulae can be applied carefully in specific situations taking these features also into account. Computer outputs in full for SNS and QL-2 theory are given and the more interesting aspects of these are briefly explained by theory.

2. Description of the test for conjunction identity

These studies arose in connection with the checking of the three conditions (a), (b), (c) under which conjunction identity is obtained for Eq.(1) / Eq.(2) which is under the condition of the SNS analogs of Eq.(36b) of MR-62 in GLOGIC. These conditions are

(a) $|P| \subseteq |Q|$ or $|Q| \subseteq |P|$, so that
$|P| \equiv |R|$ or $|Q| \equiv |R|$

(b) Wrongly given

(c) $|P| \otimes |Q| = |\underline{\underline{E}}(k, \ell)|$

It was noticed that the condition (b) as given in MR-60 is incorrect. In fact, this arose because $|Q|$ was taken to be $|E(2, 3)|$ instead of $|E(3, 2)|$, in a manually calculated result contained in page 32 of MR-60, which was the only example used for deriving this condition. The corrected matrix has now been found to satisfy the revised condition (b) discussed below. Therefore, a complete check was made of all the possible pairs $(|P|, |Q|)$, where $|P|$ and $|Q|$ are one of the 16 possible 2x2 matrices in SNS.

Although there are 16 x 16 = 256 possibilities, 16 of these correspond to |P| and |Q| being the same. The remaining 240 combinations can be made into 120 pairs, consisting of |R| = |P| ⊗ |Q| and |Q| ⊗ |P| , for which the condition for conjunction identity is the same. Therefore, all the 120 independent possibilities were checked. Of these, those for which the two ways of working out the conjunction in Eq.(1) are the same, have been picked out and listed in Table 1 below. The list contains the matrices |P| , |Q| and |R| and a check of the condition (a), and of (b) or (c) if it is not satisfied. It will be seen that condition (a) holds for a vast majority of cases and only six cases are left out. Of these, two satisfy the condition (c) corresponding to $|R| = |\underline{\underline{E}}(1, 1)|$ and $|R| = |\underline{\underline{E}}(1, 2)|$. We are left with four others and a simple inspection shows that the condition (b) can be reformulated as follows to cover these also.

(b) Both |P| and |Q| are singular matrices, with one of them having two 1's in a row and the other having two 1's in a column.

With this revision, the three conditions (a), (b), (c) are sufficient to cover all possible combinations of SNS matrices |P| and |Q| for which Eq.(1) holds. The theoretical consequences of these have been discussed in MR-62, Section 9, where it is shown that these ideas obtained for SNS logic are extendable to quantifier logic QL-2, employing 3x3 matrices, and to the general logic of relations, employing m x n matrices.

Table 1

List of all possible cases where $(\underline{a}\ \underline{P}\ \underline{b}) \wedge (\underline{a}\ \underline{Q}\ \underline{b}) = (\underline{a}\ \underline{R}\ \underline{b})$ holds for $|\underline{P}| \circledast |\underline{Q}| = |\underline{R}|$ in SNS*

$\|P\|$	$\|Q\|$	$\|R\|$	Condition (a)	Condition (b) or (c)	Number
0 0	0 0	0 0	Yes		1
0 0	0 1	0 0			
0 0	0 0	0 0	Yes		2
0 0	1 0	0 0			
0 0	0 0	0 0	Yes		3
0 0	1 1	0 0			
0 0	0 1	0 0	Yes		4
0 0	0 0	0 0			
0 0	0 1	0 0	Yes		5
0 0	0 1	0 0			
0 0	0 1	0 0	Yes		6
0 0	1 0	0 0			
0 0	0 1	0 0	Yes		7
0 0	1 1	0 0			
0 0	1 0	0 0	Yes		8
0 0	0 0	0 0			
0 0	1 0	0 0	Yes		9
0 0	0 1	0 0			
0 0	1 0	0 0	Yes		10
0 0	1 0	0 0			
0 0	1 0	0 0	Yes		11
0 0	1 1	0 0			

* If $(|P|, |Q|)$ is listed, then $(|Q|, |P|)$ is not listed, and also the cases where $|P|$ is the same as $|Q|$ are omitted.

|P|	|Q|	|R|	Condition (a)	Condition (b) or (c)	Number
0 0 0 0	1 1 0 0	0 0 0 0	Yes		12
0 0 0 0	1 1 0 1	0 0 0 0	Yes		13
0 0 0 0	1 1 1 0	0 0 0 0	Yes		14
0 0 0 0	1 1 1 1	0 0 0 0	Yes		15
0 0 0 1	0 0 1 1	0 0 0 1	Yes		17
0 0 0 1	0 1 0 1	0 0 0 1	Yes		19
0 0 0 1	1 0 0 0	0 0 0 0	Yes		21
0 0 0 1	1 0 0 1	0 0 0 1	Yes		23
0 0 0 1	1 0 1 1	0 0 0 1	Yes		25
0 0 0 1	1 1 0 1	0 0 0 1	Yes		27
0 0 0 1	1 1 1 1	0 0 0 1	Yes		29
0 0 1 0	0 0 1 1	0 0 1 0	Yes		30

|P|	|Q|	|R|	Condition(a)	Condition (b) or (c)	Number
0 0 1 0	0 1 1 0	0 0 1 0	Yes		33
0 0 1 0	0 1 1 1	0 0 1 0	Yes		34
0 0 1 0	1 0 1 0	0 0 1 0	Yes		37
0 0 1 0	1 0 1 1	0 0 1 0	Yes		38
0 0 1 0	1 1 1 0	0 0 1 0	Yes		41
0 0 1 0	1 1 1 1	0 0 1 0	Yes		42
0 0 1 1	0 1 0 1	0 0 0 1		(b) Yes	44
0 0 1 1	0 1 1 1	0 0 1 1	Yes		46
0 0 1 1	1 0 1 0	0 0 1 0		(b) Yes	49
0 0 1 1	1 0 1 1	0 0 1 1	Yes		50
0 0 1 1	1 1 1 1	0 0 1 1	Yes		54

|P|	|Q|	|R|	Condition(a)	Condition (b) or (c)	Number
0 1	0 1	0 1	Yes		55
0 0 =	0 1	0 0			
0 1	0 1	0 1	Yes		56
0 0	1 0	0 0			
0 1	0 1	0 1	Yes		57
0 0	1 1	0 0			
0 1	1 1	0 1	Yes		62
0 0	0 0	0 0			
0 1	1 1	0 1	Yes		63
0 0	0 1	0 0			
0 1	1 1	0 1	Yes		64
0 0	1 0	0 0			
0 1	1 1	0 1	Yes		65
0 0	1 1	0 0			
0 1	0 1	0 1	Yes		67
0 1	1 1	0 1			
0 1	1 1	0 1		(b) Yes	72
0 1	0 0	0 0			
0 1	1 1	0 1	Yes		73
0 1	0 1	0 1			
0 1	1 1	0 1	Yes		75
0 1	1 1	0 1			
0 1	0 1	0 1	Yes		76
1 0	1 1	1 0			

|P|	|Q|	|R|	Condition(a)	Condition (b) or (c)	Number
0 1	1 1	0 1	Yes		83
1 0	1 0	1 0			
0 1	1 1	0 1	Yes		84
1 0	1 1	1 0			
0 1	1 1	0 1		(c) Yes	91
1 1	1 0	1 0			
0 1	1 1	0 1	Yes		92
1 1	1 1	1 1			
1 0	1 0	1 0	Yes		93
0 0	0 1	0 0			
1 0	1 0	1 0	Yes		94
0 0	1 0	0 0			
1 0	1 0	1 0	Yes		95
0 0	1 1	0 0			
1 0	1 1	1 0	Yes		96
0 0	0 0	0 0			
1 0	1 1	1 0	Yes		97
0 0	0 1	0 0			
1 0	1 1	1 0	Yes		98
0 0	1 0	0 0			
1 0	1 1	1 0	Yes		99
0 0	1 1	0 0			

|P|	|Q|	|R|	Condition(a)	Condition (b) or (c)	Number
1 0 0 1	1 0 1 1	1 0 0 1	Yes		101
1 0 0 1	1 1 0 1	1 0 0 1	Yes		103
1 0 0 1	1 1 1 1	1 0 0 1	Yes		105
1 0 1 0	1 0 1 1	1 0 1 0	Yes		106
1 0 1 0	1 1 0 0	1 0 0 0		(b) Yes	107
1 0 1 0	1 1 1 0	1 0 1 0	Yes		109
1 0 1 0	1 1 1 1	1 0 1 0	Yes		110
1 0 1 1	1 1 0 1	1 0 0 1		(c) Yes	112
1 0 1 1	1 1 1 1	1 0 1 1	Yes		114
1 1 0 0	1 1 0 1	1 1 0 0	Yes		115
1 1 0 0	1 1 1 0	1 1 0 0	Yes		116

P	Q	R	Condition (a)	Condition (b) or (c)	Number
1 1 0 0	1 1 1 1	1 1 0 0	Yes		117
1 1 0 1	1 1 1 1	1 1 0 1	Yes		119
1 1 1 0	1 1 1 1	1 1 1 0	Yes		120

Total 71 examples

Thus, Table 1 has been thoroughly checked to be explainable using the three conditions (a), (b), (c). The question now arises as to why the equation given in (1) is violated in the remaining cases. For doing this, the full table of outputs for all the 120 cases was examined. The full table is not given below, but an illustrative three pages of Sl. Nos 37 to 45 is given in Table 2. Since, from the theory given in MR-62, the conjunction identity will be theoretically true, if both $\underline{\underline{a}}$ and $\underline{\underline{b}}$ are basic states (in this case T or F), in making the check of Eq.(1), not all/16 possible combinations of ($\underline{\underline{a}}$, $\underline{\underline{b}}$) the need be tested, but only 5 of them involving mixed states, namely T, D ; D, T ; F, D ; D, F ; D, D . It is to be noted that no tests need be made with $\underline{\underline{a}}$, or $\underline{\underline{b}}$, equal to X , since

Eq.(1) will always be true from theory if SSTV is used in such cases. Therefore, the possibility of generalizing Eq.(1), which is always true in classical logic, to the new states and the new algebra of SNS logic, can be tested by the data given in the full output of the type shown in Table 2. The listing in Table 2 consists of the following. For each Sl. No. we have firstly

$$\underline{\underline{a}} \quad \underline{\underline{b}} \quad \underline{\underline{c}} \quad \underline{\underline{d}} \qquad \underline{\underline{G}}(\underline{\underline{c}} \ , \underline{\underline{d}}) = \underline{\underline{g}} \tag{4}$$

where

$\underline{\underline{c}}$ is the truth value of the l.h.s of Eq.(1), and

$\underline{\underline{d}}$ is the truth value of the r.h.s of Eq.(1), and

$\underline{\underline{g}}$ gives the truth value of the agreement between the two.

For each Sl.No. a check is made whether all $\underline{\underline{g}}$'s are T, and if so, the result is printed as AGREE. If not, the disagreement is printed as DISAGREE, and the disagreeing truth values are enclosed in circles. The matrices $|P|$, $|Q|$, $|R|$, for each Sl. No. is printed at the end.

All those for which the result is AGREE have been listed in Table 1 and their characteristics have been analysed. An explanation of these, from theory, is given in Sections 3 and 4, while those, for which there are disagreements, have also similarly been collected, and are given in Table 3 in Section 5. Their theory is also given in Section 5.

Table 2: Sample of the computer output of the conjunction identity check made for Eq.(1).

```
T  D     F  F     T
F  D     D  D     T
D  T     D  D     T
D  F     F  F     T
D  D     D  D     T
                  AGREE
0  0     1  0     0  0
1  0     1  0     1  0
```

37

```
T  D     F  F     T
F  D     D  D     T
D  T     D  D     T
D  F     F  F     T
D  D     D  D     T
                  AGREE
0  0     1  0     0  0
1  0     1  1     1  0
```

38

```
T  D     F  F     T
F  D     F  F     T
D  T     (D)(F)   F
D  F     F  F     T
D  D     (D)(F)   F
                  DISA(
0  0     1  1     0
1  0     0  0     0
```

39

T	D	F	F	T
F	D	(D)	(F)	F
D	T	(D)	(F)	F
D	F	F	F	T
D	D	(D)	(F)	F

DISAGREE

0	0	1	1	0	0
1	0	0	1	0	0

40

T	D	F	F	T
F	D	D	D	T
D	T	D	D	T
D	F	F	F	T
D	D	D	D	T

AGREE

0	0	1	1	0	0
1	0	1	0	1	0

41

T	D	F	F	T
F	D	D	D	T
D	T	D	D	T
D	F	F	F	T
D	D	D	D	T

AGREE

0	0	1	1	0	0
1	0	1	1		0

42

T	D	F	F	T
F	D	F	F	T
D	T	F	F	T
D	F	D	F	F
D	D	D	F	F

DISAGREE

0	0	0	1	0 0
1	1	0	0	0 0

43

T	D	F	F	T
F	D	D	D	T
D	T	F	F	T
D	F	D	D	T
D	D	D	D	T

AGREE

0	0	0	1	0 0
1	1	0	1	0 1

44

T	D	F	F	T
F	D	D	D	T
D	T	D	D	T
D	F	D	F	F
D	D	D	D	T

DISAGREE

0	0	0	1	0 0
1	1	1	0	1 0

45

3. Logical significance of the results listed in Table 1

(a) SNS Logic

We have indicated the general need of BA-2 employed in SNS logic for describing the truth values of classical logic. In classical logic, the conjunction identity discussed here is quite generally valid and, in fact, it is equally valid in SNS if we restrict ourselves only to BA-1 truth values and use only the pure states T = (1 0) and F = (0 1) as inputs in the relation (1). However, if we include also mixed states D and X as inputs, this is no longer true. So also, for general states representable by a Boolean m-vector and Boolean n-vector in GLOGIC, the conjunction identity is not necessarily true for all cases, as was shown in MR-62. The question, now, is to find out the conditions under which it does in fact hold. This can be done by discussing the logical significance of the algebraic conditions (a), (b), (c) mentioned above. Thus, for the three conditions (a), (b), (c) in BA-2 for the conjunction identity to hold, they are equivalent to the following logical conditions:

Condition (a)

$$(\underline{a}\ \underline{\underline{P}}\ \underline{b}) \Longrightarrow (\underline{a}\ \underline{\underline{Q}}\ \underline{b}) \text{ or } \underline{\text{vice versa}}, \text{ so that} \qquad (5)$$

$$(\underline{a}\ \underline{\underline{P}}\ \underline{b}) \Longleftrightarrow (\underline{a}\ \underline{\underline{R}}\ \underline{b}) \text{ or } (\underline{a}\ \underline{\underline{Q}}\ \underline{b}) \Longleftrightarrow (\underline{a}\ \underline{\underline{R}}\ \underline{b}) \qquad (6)$$

This follows from the fact that the inclusion condition for two matrices is the same as the implication condition for the logical relations expressed by these matrices. The following two simple examples (7) and (8) illustrate this fact.

$$(\underline{a} \wedge \underline{b}) \Longrightarrow (a \Longrightarrow b) \qquad (7a)$$

with the matrices

$$|P| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad |Q| = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad |R| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad (7b)$$

$$(\underline{a} \wedge \underline{b} \text{ is true}) \Longrightarrow (\underline{a} \text{ is always true irrespective of } \underline{b}) \qquad (8a)$$

corresponding to the matrices

$$|P| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad |Q| = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \quad |R| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad (8b)$$

Thus condition (a) is equivalent to saying that either the relation $|P|$ implies relation $|Q|$ or the relation $|Q|$ implies the relation $|P|$, which appears to be quite reasonable from pure logic.

Condition (b)

A singular matrix with two 1's in a row or two 1's in a column, corresponds to one of the four relations "$\underline{\underline{a}}$ is always T(F)" , or "$\underline{\underline{b}}$ is always T(F)". Consequently, when two such relations are simultaneously present, we will get a relation of the type, "$\underline{\underline{a}}$ and $\underline{\underline{b}}$ is true" or "$\underline{\underline{a}}$ is true and $\underline{\underline{b}}$ is false" etc.

Two examples are given in (9) and (10).

($\underline{\underline{a}}$ is always true) and ($\underline{\underline{b}}$ is always true) $\Longleftrightarrow$ ($\underline{\underline{a}} \wedge \underline{\underline{b}}$ is true) (9a)

corresponding to the matrices

$$|P| = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \quad |Q| = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \quad |R| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad (9b)$$

($\underline{\underline{a}}$ is always false) and ($\underline{\underline{b}}$ is always true) $\Longleftrightarrow$

($\underline{\underline{a}}$ is false and $\underline{\underline{b}}$ is true) (10a)

corresponding to

$$|P| = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad |Q| = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \quad |R| = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \qquad (10b)$$

Condition (c)

The only two examples that are relevant under this condition correspond to the classical logical identities, namely

$$(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}) \wedge (\neg \underline{\underline{a}} \Longrightarrow \neg \underline{\underline{b}}) \Longleftrightarrow (\underline{\underline{a}} \Longleftrightarrow \underline{\underline{b}}) \qquad (11a)$$

corresponding to

$$|P| = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad |Q| = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad |R| = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad (11b)$$

and

$$(\underline{\underline{a}} \Longrightarrow \neg \underline{\underline{b}}) \wedge (\neg \underline{\underline{a}} \Longrightarrow \underline{\underline{b}}) \Longleftrightarrow (\underline{\underline{a}} \Longleftrightarrow \neg \underline{\underline{b}}) \qquad (12a)$$

corresponding to

$$|P| = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad |Q| = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \quad |R| = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad (12b)$$

It is very interesting that these classical definitions of equivalence and negation in terms of forward and reverse implications being both simultaneously present, can be extended without change of form completely to SNS logic for all its states.

The above discussion shows that the conditions (a), (b) and (c) are readily understandable from classical logic. What is, however, not so obvious is the fact that these are the only three independent possibilities for which classical logic is extendable to SNS logic for all possible inputs ($\underline{\underline{a}}, \underline{\underline{b}}$), with $\underline{\underline{a}}$, $\underline{\underline{b}}$ being products of relations in T, F, D. It does not appear to be a simple matter to do this by pure logical reasoning. However, the use of 2x2 Boolean matrices for all possible relations in propositional calculus

and the Boolean algebraic fact that there are only 16 of these, make it possible to prove that no other conditions need be examined for verifying the validity of the conjunction identity. On the other hand, as mentioned in MR-62(see below) / , the disjunction identity is true for all 2x2 matrices $|P|$ and $|Q|$, as it is only a special case of all mxn matrices.

(b) GLOGIC

We shall first give the formulae corresponding to the conjunction non-identity and disjunction identity respectively for the Boolean truth values of two relations in GLOGIC.

Conjunction non-identity

$$\langle \underset{\sim}{a} | \underset{\sim}{P} | \underset{\sim}{b} \rangle \otimes \langle \underset{\sim}{a} | \underset{\sim}{Q} | \underset{\sim}{b} \rangle \neq \langle \underset{\sim}{a} | \underset{\sim}{R} | \underset{\sim}{b} \rangle \quad \text{in general} \tag{13a}$$

where

$$|\underset{\sim}{P}| \circledast |\underset{\sim}{Q}| = |\underset{\sim}{R}| \tag{13b}$$

Disjunction identity

$$\langle \underset{\sim}{a} | \underset{\sim}{P} | \underset{\sim}{b} \rangle \oplus \langle \underset{\sim}{a} | \underset{\sim}{Q} | \underset{\sim}{b} \rangle \iff \langle \underset{\sim}{a} | \underset{\sim}{R} | \underset{\sim}{b} \rangle \tag{14a}$$

$$\text{for} \quad |\underset{\sim}{P}| \circledast |\underset{\sim}{Q}| = |\underset{\sim}{R}| \tag{14b}$$

The MATLOG function which is relevant for (13) and (14) is the general Boolean truth value GBTV, and Eqs.(13) and (14) can be written in MATLOG notation as below.

GBTV(GVA, GMP, GVB) = BP (15a)

GBTV(GVA, GMQ, GVB) = BQ (15b)

GBTV(GVA, GMR, GVB) = BR (15c)

Conjunction non-identity

Then the conjunction non-identity takes the form

GMXPDT(GMP, GMQ) = GMR (16a)

BPDT(BP, BQ) = BRP (16b)

BRP $\neq$ BR (May or may not be equal) (17)

Disjunction identity

GMXSUM(GMP, GMQ) = GMR (18a)

BSUM(BP, BQ) = BRPP (18b)

BRPP $\Longleftrightarrow$ BR (Always true) (19)

The algebraic proof, of the non-identity of the r.h.s and l.h.s of Eq.(3a) in general, has been shown for GLOGIC in MR-62 (Part III) and it has been discussed in good detail there. As mentioned above, since SNS is a particular case of GLOGIC, the same result should hold for SNS also for binary truth values of the type SBTV. The corresponding examples in QL-2 are not discussed since they only confirm the general discussion made here, for the particular case of GLOGIC with MI = 3, MJ = 3. However, it is

interesting to consider the corresponding non-identity for unary relations in GLOGIC, particularly for the conjunction non-identity. We shall illustrate this by considering Eqs. (20), (21) below, which are analogous to (16) and (17) for binary truth values.

$$\text{GUNPDT(GVA, GMP)} = \text{GVBP} \tag{20a}$$

$$\text{GUNPDT(GVA, GMQ)} = \text{GVBQ} \tag{20b}$$

$$\text{GUNPDT(GVA, GMR)} = \text{GVC} \tag{20c}$$

$$\text{GMXPDT(GMP, GMQ)} = \text{GMR} \tag{21a}$$

$$\text{GVIDYA(GVBP, GVBQ)} = \text{GVD} \tag{21b}$$

$$\neq \text{GVC (in general)} \tag{21c}$$

The unary and binary conjunction non-identities for two relations in GLOGIC are related to one another as indicated in the next Section 4(a). We shall note here this relationship in terms of MATLOG notation as in (22a,b,c) below,

$$\text{GSCPDT(GVBP, GVB)} = \text{BP} \tag{22a}$$

$$\text{GSCPDT(GVBQ, GVB)} = \text{BQ} \tag{22b}$$

$$\text{GSCPDT(GVBR, GVB)} = \text{BR} \tag{22c}$$

when it turns out that the condition that GVC is not equal to GVD in general, of (21), implies BRP not equal to BR, in general, of (17) We shall consider unary relations more in detail in the next section.

4. Failure of conjunction identity for two relations in GLOGIC

(a) Venn diagram treatment of binary relation

We seek to verify whether the non-equivalence in (23) occurs for three sets A, B, C in general.

$$(A \subseteq C) \wedge (B \subseteq C) \not\Longleftrightarrow (A \cap B) \subseteq C \qquad (23)$$

Figs. 1(a), and 1(b), are two Venn diagrams showing examples of A, B, C for which the identity (23) is true, and is not true. The figures are self-explanatory. It is readily verified that, in both cases, the forward implication in (23), expressed as (24a), is always true, while the reverse implication, expressed as (24b) is true for Fig. 1(a), while it is not so for Fig. 1(b).

$$(A \subseteq C) \wedge (B \subseteq C) \Longrightarrow (A \cap B) \subseteq C \qquad (24a)$$

$$(A \cap B) \subseteq C \not\Longrightarrow (A \subseteq C) \wedge (B \subseteq C) \qquad (24b)$$

Thus, the truth values of the set-theoretical statements in the l.h.s and r.h.s of (23) need not necessarily be the same.

On the other hand, for the disjunction identity, Eq.(25) is always true, as there is only one type of Venn diagram, namely that shown in Fig. 2. In this case, the implication holds both in the forward direction and reverse direction, and (25) is valid as an equivalence.

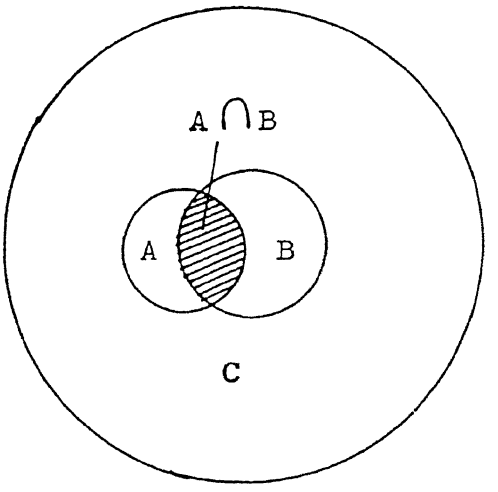


Fig. 1(a)

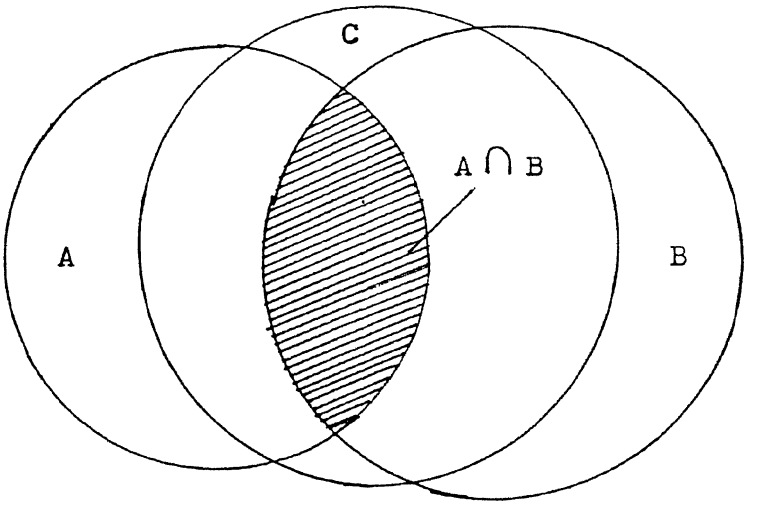


Fig. 1(b)

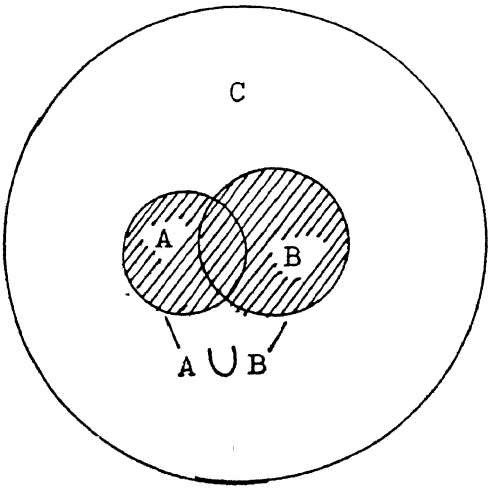


Fig.2

We shall give some examples of the tests in Eqs. (22) and (25) in Section 4(b), after discussing examples of non-equivalence of unary relations as expressed in (19) to (21). The occurrence of this feature for set theory does not seem to have been specifically pointed out. However, it is important to recognize the occurrence of conjunction non-identity, which becomes quite clear when the l.h.s and r.h.s are formulated in MATLOG language.

(b) Test of unary form of conjunction non-identity

The example given below is taken from MR-64 which was an elementary presentation of the capabilities of BVMF for the theory of relations. Using the notation given in this report, the two relations are

$$\underset{\sim}{P} \; : \quad b_j \text{ is the professor of } a_i \tag{26a}$$

$$\underset{\sim}{Q} \; : \quad b_j \text{ is the examiner of } a_i \tag{26b}$$

The corresponding matrices are

$$GMP = \underset{\sim}{P} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \quad GMQ = \underset{\sim}{Q} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \tag{27}$$

The matrix $\underset{\sim}{R}$, represented in MATLOG by GMR, given by (19), corresponds to the property that b_j is both "professor of" as well as "the examiner of" the student a_i. We are not giving the matrix GMR, as the program will take care of this.

Thus the inputs to the problem are

a_i corresponds to the students a_1 to a_5

b_j corresponds to the professors b_1 to b_4

The two relations considered are representable by the matrices GMP, GMQ, while GMR, which is the Boolean product of the matrices GMP and GMQ, representing the conjunction of the two relations. With these, two questions can be asked, as in (a), (b) below, which will lead to the answers GVC and GVD, respectively, of Eqs. (20c) and (21).

(a) Find those professors b_j (GVC) who are both "professor of" as well as "examiner of" the students a_2 and a_3.

(b) Find those professors b_j (GVD) who teach at least one of the students a_2, a_3 and also examine either a_2 or a_3.

Thus, the properties of being "professor of" and of being "examiner of" are the entities that are combined by conjunction in problem (a) for the same professor. On the other hand, it is the set of professors who have the property of being "professor of" and the set having the property of being "examiner of", some member of the given set of students, that are combined by conjunction in problem (b). The distinction between the two can be indicated quite clearly in MATLOG by the statements (28) and (29) below :

GVC = GUNPDT(GVA, GMXPDT(GMP,GMQ)) (28)

GVD = GVIDYA(GUNPDT(GVA, GMP), GUNPDT(GVA, GMQ)) (29)

It is quite clear that the two are entirely different functions of the inputs GVA, GMP, GMQ. On the other hand, it is very difficult to state this distinction in simple terms using the notation of standard logic or that of clausal relations. (in fact, we found difficulty in even expressing it without using BVMF concepts and notation.)

Consequently, the answers to the two questions can be given as follows:

Inputs : GVA = (0 1 1 0 0), and GMP, GMQ of (27) (30)

Outputs: (a) GVC = (0 0 0 1), (b) GVD = (1 1 0 1) (31)

Thus, we find that the two sets of professors given by GVC and GVD are quite different. The former, satisfying the conditions of problem (a), consists only of b_4, while the latter, satisfying the condition of problem (b), consists of b_1, b_2 and b_4. What is even more interesting is that the latter includes the former, thus exhibiting the implication property of the conjunction identity indicated by Eq. (23). This particular problem does not have all the features that are relevant for the Venn diagrams in Figs 1(a) and (b), but are particular cases of the sets A, B, C considered in Section 5(a) with reference to these figures. It is satisfying to know that formal logic, formal set theory, and BVMF and MATLOG, are all inter-related this way and are mutually consistent with one another.

In the next section, we shall briefly consider the relation between the conjunction non-identity for unary relations given in (19) to (21), with the corresponding property for binary relations given in Eqs. (15) to (17).

(c) Relation between conjunction non-identities occurring in GLOGIC, QLOGIC and SNS

In the most general case of GLOGIC, the conjunction non-identity is expressible for binary relations by Eqs. (15), (16), (17), leading to the non-identity of the product of the Boolean truth values of two relations with the Boolean truth value of the product of the two relations, as given in (17). The corresponding result for the outputs of unary relations, as given in (19) to (21), is that the conjunction of the outputs of two relations (namely the vidya product) is not equal to the output of the conjunction of the two relations (namely their Boolean product). The unary and binary aspects are, as mentioned above, closely related, in that the Boolean truth value GBTV of $\underline{\underline{R}}(\underline{\underline{a}}, \underline{\underline{b}})$ is given by the binary product in (32):

$$t(\underline{\underline{a}}\ \underline{\underline{R}}\ \underline{\underline{b}}) = \langle a | R | b \rangle \qquad (32)$$

which is also the scalar product of the unary product $\langle a|R|$ and the vector $|b\rangle$. Hence, if the unary products GVC and GVD of (21) are different, then the truth values BR and BRP of (17) also need not be equal. This is specifically mentioned here because the unary product is the one that is made use of in the

clausal form of logic (as in the Type-1 clausal relations discussed in MR-57). In fact, the distinction between the nature of the two Venn diagrams, of conjunction and disjunction of two relations, as shown in Figs. (1) and (2), came to our attention in a discussion that the author had with some colleagues regarding possible combinations of two clausal relations to yield a third one, as is considered in Ref(1), page 5. The two examples given in Ref[1] are as in (33a) and (33b), whose nature in BVMF notation is also indicated below :

Disjunction of two relations

Mother (x, y) or father (x, y) implies parent (x, y) (33a)

One type of tensor product of two relations

Mother (x, z) and father (y, z) implies parents (x, y, z) (33b)

An examination of all possible ways in which two relations can be combined, led us to the fact that disjunction and conjunctio of two relations have different types of Venn diagrams, which looked puzzling at that time. However, the treatment given here, in terms of MATLOG as shown above, has indicated that the structure presented in Section 4(a) of the conjunction non-identity in Eq.(23), namely of its being only an implication in the forward direction as in Eq.(24b), is reasonable and practically applicable in terms of MATLOG.

5. Conditions for conjunction non-identity in SNS

The previous Section 4 dealt with a discussion of conditions under which Eq.(1) for conjunction identity is valid and all the independent examples in SNS are listed in Table 1. We now consider the case of conjunction non-identity, i.e. when Eq.(1) is definitely false, and list all such (independent) cases from the full data of the type shown in Table 2. This has been done in Table 3, and some of the rules governing these, and regularities that are observed, will be discussed theoretically in this section.

The first feature to be noticed in Table 2 is that the non-identity, when it occurs, always has D for the l.h.s of Eq.(1) and F for the r.h.s. This is not only true for the examples of DISAGREE shown in Table 2, but is true for every example of DISAGREE listed in Table 3.

This feature can be given a simple explanation from BVMF, by examining the nature of the differences between the component Boolean truth values (SBTV) of the SNS truth values (SSTV) $\underline{c}$ and $\underline{d}$ for the l.h.s and r.h.s of Eq.(1). We shall show that (c_1,d_1) can have only the values (1, 0) and that (c_2,d_2) can have only the values (1, 1) so that $\underline{c} = D$ and $\underline{d} = F$ whenever there is conjunction non-identity for Eq.(1).

Table 3 :

List of all different cases* where $(\underline{a}\ P\ \underline{b}) \wedge (\underline{a}\ Q\ \underline{b}) \not\Rightarrow (\underline{a}\ R\ \underline{b})$ †

Sl.No.	Inputs $\underline{a}$ & $\underline{b}$			Matrices |P|	|Q|	|R|
16	F,D	D,D		0 0 0 1	0 0 1 0	0 0 0 0
18	D,F	D,D		0 0 0 1	0 1 0 0	0 0 0 0
20	F,D	D,F	D,D	0 0 0 1	0 1 1 0	0 0 0 0
22	D,D			0 0 0 1	1 0 0 0	0 0 0 0
24	F,D	D,D		0 0 0 1	1 0 1 0	0 0 0 0
26	D,F	D,D		0 0 0 1	1 1 0 0	0 0 0 0
28	F,D	D,F	D.D	0 0 0 1	1 1 1 0	0 0 0 0
31	D,D			0 0 1 0	0 1 0 0	0 0 0 0
32	F,D	D,D		0 0 1 0	0 1 0 1	0 0 0 0
35	D,T	D,D		0 0 1 0	1 0 0 0	0 0 0 0

* If (P , Q) is listed, then (Q , P) is not listed.

† In all cases l.h.s is D and r.h.s is F.

Sl. No.	Inputs			Matrices
36	F,D	D,T	D,D	0 0 1 0 0 0 1 0 0 1 0 0
39	D,T	D,D		0 0 1 1 0 0 1 0 0 0 0 0
40	F,D	D,T	D,D	0 0 1 1 0 0 1 0 0 1 0 0
43	D,F	D,D		0 0 0 1 0 0 1 1 0 0 0 0
45	D,F			0 0 0 1 0 0 1 1 1 0 1 0
47	D,T	D,D		0 0 1 0 0 0 1 1 0 0 0 0
48	D,T			0 0 1 0 0 0 1 1 0 1 0 1
51	D,T	D,F	D,D	0 0 1 1 0 0 1 1 0 0 0 0
52	D,T			0 0 1 1 0 0 1 1 0 1 0 1
53	D,F			0 0 1 1 0 0 1 1 1 0 1 0

Sl.No.	Inputs			Matrices
58	T,D	D,D		0 1 1 0 0 0 0 0 0 0 0 0
59	T,D	D,F	D,D	0 1 1 0 0 0 0 0 0 1 0 0
60	T,D	D,D		0 1 1 0 0 0 0 0 1 0 0 0
61	T,D	D,F	D,D	0 1 1 0 0 0 0 0 1 1 0 0
66	F,D			0 1 0 1 0 1 0 1 1 0 0 0
68	T,D	D,D		0 1 1 0 0 0 0 1 0 0 0 0
69	T,D			0 1 1 0 0 0 0 1 0 1 0 1
70	T,D	F,D	D,D	0 1 1 0 0 0 0 1 1 0 0 0
71	T,D			0 1 1 0 0 0 0 1 1 1 0 1
74	F,D			0 1 1 1 0 1 0 1 1 0 0 0

Sl.No.	Inputs	Matrices
77	T,D D,T D,D	0 1 1 0 0 0 1 0 0 0 0 0
78	T,D F,D D,T D,F D,D	0 1 1 0 0 0 1 0 0 1 0 0
79	T,D	0 1 1 0 0 0 1 0 1 0 1 0
80	T,D D,F	0 1 1 0 0 0 1 0 1 1 1 0
81	D,T	0 1 1 1 0 1 1 0 0 0 0 0
82	F,D D,T	0 1 1 1 0 1 1 0 0 1 0 0
85	T,D D,T D,D	0 1 1 0 0 0 1 1 0 0 0 0
86	T,D D,T	0 1 1 0 0 0 1 1 0 1 0 1
87	T,D	0 1 1 0 0 0 1 1 1 0 1 0
88	T,D	0 1 1 0 0 0 1 1 1 1 1 1

Sl. No.	Inputs		Matrices
89	D,T		0 1 1 1 0 1 1 1 0 0 0 0
90	D,T		0 1 1 1 0 1 1 1 0 1 0 1
100	F,D		1 0 1 0 1 0 0 1 1 0 0 0
102	D,F		1 0 1 1 1 0 0 1 0 0 0 0
104	F,D	D,F	1 0 1 1 1 0 0 1 1 0 0 0
108	F,D		1 0 1 1 1 0 1 0 0 1 0 0
111	D,F		1 0 1 1 1 0 1 1 0 0 0 0
113	D,F		1 0 1 1 1 0 1 1 1 0 1 0
118	F,D		1 1 1 1 1 1 0 1 1 0 0 0

Total 49 examples

Expressed in terms of Boolean truth values, Eq.(1) can be rewritten as the two equations, in (33a) and (33b):

$$\langle \underline{a}|\underline{P}|\underline{b}\rangle \otimes \langle \underline{a}|\underline{Q}|\underline{b}\rangle = \langle \underline{a}|\underline{R}|\underline{b}\rangle \qquad (33a)$$

$$\langle \underline{a}|\underline{P}^c|\underline{b}\rangle \oplus \langle \underline{a}|\underline{Q}^c|\underline{b}\rangle = \langle \underline{a}|\underline{R}^c|\underline{b}\rangle \qquad (33b)$$

In the former, the condition of Eq.(2), namely

$$|\underline{P}| \circledast |\underline{Q}| = |\underline{R}| \qquad (34a)$$

is assumed to hold. It is then obvious that, in (33b), the condition

$$|\underline{P}^c| \circledoplus |\underline{Q}^c| = |\underline{R}^c| \qquad (34b)$$

is necessarily satisfied. Under this latter condition, the second Eq.(33b) is always true, being a disjunction identity which is universally valid, so that $c_2 = d_2$. Therefore, we need consider only Eq.(33a) for the Boolean truth values c_1, d_1 to check for the occurrence of conjunction non-identity between the SNS truth values $\underline{c}$ and $\underline{d}$.

As already mentioned in connection with Table 2, non-identity between the truth values $\underline{c}$ and $\underline{d}$ of the l.h.s and r.h.s of Eq.(1) can occur only for combinations of $\underline{a}$ and $\underline{b}$ for which one of them has the doubtful state D and the other has T, F, D, but not X. Hence it follows that, *if* $d_1 = \langle \underline{a}|\underline{R}|\underline{b}\rangle = 1$, *then*, since $|\underline{R}|$ is the Boolean product of $|\underline{P}|$ and $|\underline{Q}|$, both

$\langle \underline{a}|\underline{P}|\underline{b}\rangle$ and $\langle \underline{a}|\underline{Q}|\underline{b}\rangle$ are also equal to 1, so that $c_1 = 1$ and there is no non-equality for these Boolean truth values c_1 and d_1 of the l.h.s and r.h.s of Eq.(1). Therefore, if there is non-identity, d_1 is necessarily 0 and we can only have the combination $d_1 = 0$ and $c_1 = 1$ but not in the reverse way.

Also, since $c_2 = d_2$ for the two sides of Eq.(33b), we can have the combination $c_2 = 1$, $d_2 = 1$, or $c_2 = 0$, $d_2 = 0$. We have seen just now that d_1 is necessarily 0, so that, if d_2 also is equal to 0, the output $\underline{d}$ is (0 0) = X. However, for any SNS binary relation obtained with the permissible truth values T, F, D as input, the output can only be again T, F or D, but not X (if the relation itself is not a null matrix). Therefore, $\underline{d}$ = X cannot occur when there is conjunction non-identity between $\underline{c}$ and $\underline{d}$, which leaves us with the only possibility

$$c_1 = 1, \quad c_2 = 1, \quad \underline{c} = D \tag{35a}$$

$$d_1 = 0, \quad d_2 = 1, \quad \underline{d} = F \tag{35b}$$

Thus, we see why all the non-identities that are listed in Table 3 are of the type (D, F) for ($\underline{c}$, $\underline{d}$).

In addition to this general feature, several regularities can be detected in the data contained in Table 3. Some of the interesting ones are listed below. They are not exhaustive, but they are given mainly for the sake of indicating how BVMF

can explain all such features. Some of them are explained below and the others are left for being worked out by the reader.

(a) If $|R| = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$, then the list of inputs $(\underline{a}, \underline{b})$ for that serial number always contains (D, D).

(b) If $|R| \neq \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$, then the corresponding list of $(\underline{a}, \underline{b})$ does not contain (D, D).

(c) If $|P| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, $|Q| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$ or

$|P| = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, $|Q| = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$, then

the list of $(\underline{a}, \underline{b})$ contains only (D, D).

(d) If $|P| = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $|Q| = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, then the list $(\underline{a}, \underline{b})$ contains all five possibilities

(T, D), (F, D), (D, T), (D, F), (D, D).

We shall explain (a), (b) and (d) below.

(a) We indicate the matrix $\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$ by $|X|$. Making use of the argument given above, it is clear that since the input matrices $|P|$ and $|Q|$ are not equal to $|X|$, then for the

input (D, D), the Boolean truth value $c_1 = 1$. Also since $|R| = |X|$, the Boolean truth value $d_1 = 0$.

Hence (D, D) always occurs if $|R| = |X|$.

(b) For the same reason if $|R|$ is not equal to $|X|$, $d_1 = 1$ and $c_1 = d_1$ so that there is no conjunction non-identity. Consequently, the list of $(\underline{a}, \underline{b})$ will not contain (D, D).

(d) Under the given conditions for $|P|$ and $|Q|$ if one of the inputs $\underline{a}$ or $\underline{b}$ is D, then the truth value of the l.h.s of Eq.(1) is $c_1 = 1$. At the same time $|R| = |X|$ so that $d_1 = 0$, thus leading to all five possibilities contained in the list of $(\underline{a}, \underline{b})$ occurring for that serial number.

6. Distinction between pure BVMF and BVMF implementation of classical logic

This section is a sequel to to Section 10 of Part III (MR-62) regarding the similarity and differences of the two ways of calculating truth values in general — namely via BVMF completely (GSTV), and via the BVMF implementation of logical relations (GSTV2). Some examples of the differences between the two have been pointed out for SNS logic in Part I (between SSTV and SSTV2, and between SUNPDT and SUNPT2)

and in quantifier logic (between QSTV and QSTV2 and QUNPDT and QUNPT2). The individual features outlined in Parts I and II have been explained in a general way by using GLOGIC formalism in Part III. It was mentioned there that more detailed tables for SNS logic and quantifier logic (QL-2) will be given in this report. This is done below, and it appears as if very little is needed by way of theoretical discussion to supplement the presentation in MR-62.

We give below in Table 4 the complete output of Problem 4C, which has the same pattern as in Table 5 of MR-62, and in Table 5 the output of Problem 8C, which also has the same format as in Table 6 of MR-62. The main features to be noted are the following.

(a) Comments on Table 4 for SNS logic

In Problem 4C, most of the truth values are equivalent for all three ways of calculating it. For the few cases where they differ, the output of $\underline{a} \; \underline{\underline{E}} \; \underline{b}$ is always F when calculated via (51a) of MR-62, and T for the output calculated via (51b), while it is X for the pure BVMF truth value. The reason for this has been already given. Essentially it is

Table 4: Full printout of the data in Table 5 of MR-62

PROB 4C (K,L) = (1,1)

T	T	T		
T	F	F		
T	D	D		
T	X	X		
F	T	F		
F	F	T		
F	D	D		
F	X	X		
D	T	D		
D	F	D		
D	D	D		
D	X	X	F	T
X	T	X		
X	F	X		
X	D	X	F	T
X	X	X		

PROB 4C (K,L) = (1,2)

T	T	F		
T	F	T		
T	D	D		
T	X	X		
F	T	T		
F	F	F		
F	D	D		
F	X	X		
D	T	D		
D	F	D		
D	D	D		
D	X	X	F	T
X	T	X		
X	F	X		
X	D	X	F	T
X	X	X		

PROB 4C (K,L) = (1,3)

T	T	T		
T	F	T		
T	D	T		
T	X	X		
F	T	F		
F	F	F		
F	D	F		
F	X	X		
D	T	D		
D	F	D		
D	D	D		
D	X	X	F	T
X	T	X		
X	F	X		
X	D	X		
X	X	X		

PROB 4C (K,L) = (1,4)

T	T	F		
T	F	F		
T	D	F		
T	X	X		
F	T	T		
F	F	T		
F	D	T		
F	X	X		
D	T	D		
D	F	D		
D	D	D		
D	X	X	F	T
X	T	X		
X	F	X		
X	D	X		
X	X	X		

PROB 4C (K,L) = (2,1)

T	T	F	
T	F	T	
T	D	D	
T	X	X	
F	T	T	
F	F	F	
F	D	D	
F	X	X	
D	T	D	
D	F	D	
D	D	D	
D	X	X	F T
X	T	X	
X	F	X	
X	D	X	F T
X	X	X	

PROB 4C (K,L) = (2,2)

T	T	T	
T	F	F	
T	D	D	
T	X	X	
F	T	F	
F	F	T	
F	D	D	
F	X	X	
D	T	D	
D	F	D	
D	D	D	
D	X	X	F T
X	T	X	
X	F	X	
X	D	X	F T
X	X	X	

PROB 4C (K,L) = (2,3)

T	T	F	
T	F	F	
T	D	F	
T	X	X	
F	T	T	
F	F	T	
F	D	T	
F	X	X	
D	T	D	
D	F	D	
D	D	D	
D	X	X	F T
X	T	X	
X	F	X	
X	D	X	
X	X	X	

PROB 4C (K,L) = (2,4)

T	T	T	
T	F	T	
T	D	T	
T	X	X	
F	T	F	
F	F	F	
F	D	F	
F	X	X	
D	T	D	
D	F	D	
D	D	D	
D	X	X	F T
X	T	X	
X	F	X	
X	D	X	
X	X	X	

```
      PROB 4C   (K,L) = (3,1)
T   T      T
T   F      F
T   D      D
T   X      X
F   T      T
F   F      F
F   D      D
F   X      X
D   T      T
D   F      F
D   D      D
D   X      X
X   T      X
X   F      X
X   D      X   F   T
X   X      X
```

```
      PROB 4C   (K,L) = (3,2)
T   T      F
T   F      T
T   D      D
T   X      X
F   T      F
F   F      T
F   D      D
F   X      X
D   T      F
D   F      T
D   D      D
D   X      X
X   T      X
X   F      X
X   D      X   F   T
X   X      X
```

```
      PROB 4C   (K,L) = (3,3)
T   T      T
T   F      T
T   D      T
T   X      X
F   T      T
F   F      T
F   D      T
F   X      X
D   T      T
D   F      T
D   D      T
D   X      X
X   T      X
X   F      X
X   D      X
X   X      X
```

```
      PROB 4C   (K,L) = (3,4)
T   T      F
T   F      F
T   D      F
T   X      X
F   T      F
F   F      F
F   D      F
F   X      X
D   T      F
D   F      F
D   D      F
D   X      X
X   T      X
X   F      X
X   D      X
X   X      X
```

```
     PROB 4C   (K,L) = (4,1)
T    T         F
T    F         T
T    D         D
T    X         X
F    T         F
F    F         T
F    D         D
F    X         X
D    T         F
D    F         T
D    D         D
D    X         X
X    T         X
X    F         X
X    D         X   F   T
X    X         X
```

```
     PROB 4C   (K,L) = (4,2)
T    T         T
T    F         F
T    D         D
T    X         X
F    T         T
F    F         F
F    D         D
F    X         X
D    T         T
D    F         F
D    D         D
D    X         X
X    T         X
X    F         X
X    D         X   F   T
X    X         X
```

```
     PROB 4C   (K,L) = (4,3)
T    T         F
T    F         F
T    D         F
T    X         X
F    T         F
F    F         F
F    D         F
F    X         X
D    T         F
D    F         F
D    D         F
D    X         X
X    T         X
X    F         X
X    D         X
X    X         X
```

```
     PROB 4C   (K,L) = (4,4)
T    T         T
T    F         T
T    D         T
T    X         X
F    T         T
F    F         T
F    D         T
F    X         X
D    T         T
D    F         T
D    D         T
D    X         X
X    T         X
X    F         X
X    D         X
X    X         X
TT2   --   STOP
```

based on the fact that the Boolean truth value of $\underline{\underline{a}} \vee \underline{\underline{b}} = \underline{\underline{t}}$ is 0 for BVMF (SBTV), and 1 for SBTV2, if one of the two inputs $\underline{\underline{a}}$ and $\underline{\underline{b}}$ is X and the other is T. It can then be shown that for the quantities listed in Table 4 above, the differences will occur only if the SNS relative truth value (SRELTV) $\underline{\underline{t}}$ in (36a) or (36b) below is D .

$$\underline{\underline{t}}(\underline{\underline{a}}\ ,\ s(k)) = D\ ,\quad \underline{\underline{t}}(\underline{\underline{b}}\ ,\ s(\ell)) = D \qquad (36a,b)$$

This explains the occurrence of the difference for the pair of inputs (D, X) for the relations $\underline{\underline{E}}(1, 1)$, $\underline{\underline{E}}(1, 2)$, $\underline{\underline{E}}(1, 3)$ $\underline{\underline{E}}(1, 4)$ for which $\underline{\underline{t}}(D, T) = D$ and for $\underline{\underline{E}}(2, 1)$, $\underline{\underline{E}}(2, 2)$, $\underline{\underline{E}}(2, 3)$, $\underline{\underline{E}}(2, 4)$ for which $\underline{\underline{t}}(D, F) = D$. On the other hand, no differences occur for $\underline{\underline{E}}(3, 3)$, $\underline{\underline{E}}(3, 4)$, $\underline{\underline{E}}(4, 3)$ and $\underline{\underline{E}}(4, 4)$ since $\underline{\underline{t}}(D, D) = T$, $\underline{\underline{t}}(D, X) = F$ and $\underline{\underline{t}}(X, D) = \underline{\underline{t}}(X, X) = X$, all of which are not equal to D . These results are very relevant for the discussion of the corresponding similar cases of QL-2 relations.

(To p. 59)

Table 5: Full printout of the data in Table 6 of MR-62

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	F	F	D	F	D	D	X
NAL	F	T	T	D	T	D	D	X
SOM	F	T	T	D	T	D	D	X
AON	D	D	D	D	D	D	D	X
NEX	F	T	T	D	T	D	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	F	F	D	F	D	D	X
NAL	F	T	T	D	T	D	D	X
SOM	F	T	T	D	T	D	D	X
AON	D	D	D	D	D	D	D	(F)
NEX	F	T	T	D	T	D	D	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	(F)	X	(F)	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	F	F	D	F	D	D	X
NAL	F	T	T	D	T	D	D	X
SOM	F	T	T	D	T	D	D	X
AON	D	D	D	D	D	D	D	(T)
NEX	F	T	T	D	T	D	D	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	(T)	X	(T)	(T)	X

1

PROB 8C (K,L) = (1,2)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	T	T	D	T	D	D	X
NAL	T	F	F	D	F	D	D	X
SOM	T	F	F	D	F	D	D	X
AON	D	D	D	D	D	D	D	X
NEX	T	F	F	D	F	D	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	T	T	D	T	D	D	X
NAL	T	F	F	D	F	D	D	X
SOM	T	F	F	D	F	D	D	X
AON	D	D	D	D	D	D	D	(F)
NEX	T	F	F	D	F	D	D	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	(F)	X	(F)	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	T	T	D	T	D	D	X
NAL	T	F	F	D	F	D	D	X
SOM	T	F	F	D	F	D	D	X
AON	D	D	D	D	D	D	D	(T)
NEX	T	F	F	D	F	D	D	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	(T)	X	(T)	(T)	X

2

PROB 8C (K,L) = (1,3)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	T	F	F	D	D	X
NAL	T	D	F	T	T	D	D	X
SOM	T	D	F	T	T	D	D	X
AON	D	D	D	D	D	D	D	X
NEX	T	D	F	T	T	D	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	T	F	F	D	D	X
NAL	T	D	F	T	T	D	D	X
SOM	T	D	F	T	T	D	D	X
AON	D	D	D	D	D	D	D	(F)
NEX	T	D	F	T	T	D	D	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	(F)	X	X	X	(F)	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	T	F	F	D	D	X
NAL	T	D	F	T	T	D	D	X
SOM	T	D	F	T	T	D	D	X
AON	D	D	D	D	D	D	D	(T)
NEX	T	D	F	T	T	D	D	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	(T)	X	X	X	(T)	(T)	X

3

ALL	T	D	F	T	T	D	D	X	T	D	F	T	T	D	D	X	T	D	F	T	T	D	D	X
NAL	F	D	T	F	F	D	D	X	F	D	T	F	F	D	D	X	F	D	T	F	F	D	D	X
SOM	F	D	T	F	F	D	D	X	F	D	T	F	F	D	D	X	F	D	T	F	F	D	D	X
AON	D	D	D	D	D	D	D	X	D	D	D	D	D	D	D	(F)	D	D	D	D	D	D	D	(T)
NEX	F	D	T	F	F	D	D	X	F	D	T	F	F	D	D	X	F	D	T	F	F	D	D	X
EXS	D	D	D	D	D	D	D	X	D	D	D	D	D	D	D	(F)	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	X	D	D	D	D	D	D	D	(F)	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X	X	(F)	X	X	X	(F)	(F)	X	X	(T)	X	X	X	(T)	(T)	X

PROB 8C (K,L) = (1,5)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	F	D	T	F	D	X	F	D	F	D	T	F	D	X	F	D	F	D	T	F	D	X
NAL	T	D	T	D	F	T	D	X	T	D	T	D	F	T	D	X	T	D	T	D	F	T	D	X
SOM	T	D	T	D	F	T	D	X	T	D	T	D	F	T	D	X	T	D	T	D	F	T	D	X
AON	D	D	D	D	D	D	D	X	D	D	D	D	D	D	D	(F)	D	D	D	D	D	D	D	(T)
NEX	T	D	T	D	F	T	D	X	T	D	T	D	F	T	D	X	T	D	T	D	F	T	D	X
EXS	D	D	D	D	D	D	D	X	D	D	D	D	D	D	D	(F)	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	X	D	D	D	D	D	D	D	(F)	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X	X	(F)	X	(F)	X	X	(F)	X	X	(T)	X	(T)	X	X	(T)	X

PROB 8C (K,L) = (1,6)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X	T	D	T	D	F	T	D	X	T	D	T	D	F	T	D	X
NAL	F	D	F	D	T	F	D	X	F	D	F	D	T	F	D	X	F	D	F	D	T	F	D	X
SOM	F	D	F	D	T	F	D	X	F	D	F	D	T	F	D	X	F	D	F	D	T	F	D	X
AON	D	D	D	D	D	D	D	X	D	D	D	D	D	D	D	(F)	D	D	D	D	D	D	D	(T)
NEX	F	D	F	D	T	F	D	X	F	D	F	D	T	F	D	X	F	D	F	D	T	F	D	X
EXS	D	D	D	D	D	D	D	X	D	D	D	D	D	D	D	(F)	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	X	D	D	D	D	D	D	D	(F)	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X	X	(F)	X	(F)	X	X	(F)	X	X	(T)	X	(T)	X	Y	(T)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	F	F	F	F	F	F	F	X
SOM	F	F	F	F	F	F	F	X
AON	D	D	D	D	D	D	D	X
NEX	F	F	F	F	F	F	F	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	F	F	F	F	F	F	F	X
SOM	F	F	F	F	F	F	F	X
AON	D	D	D	D	D	D	D	(F)
NEX	F	F	F	F	F	F	F	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	F	F	F	F	F	F	F	X
SOM	F	F	F	F	F	F	F	X
AON	D	D	D	D	D	D	D	(T)
NEX	F	F	F	F	F	F	F	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X

PROB 8C (K,L) = (1,8)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	T	T	T	T	T	T	T	X
SOM	T	T	T	T	T	T	T	X
AON	D	D	D	D	D	D	D	X
NEX	T	T	T	T	T	T	T	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	T	T	T	T	T	T	T	X
SOM	T	T	T	T	T	T	T	X
AON	D	D	D	D	D	D	D	(F)
NEX	T	T	T	T	T	T	T	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	T	T	T	T	T	T	T	X
SOM	T	T	T	T	T	T	T	X
AON	D	D	D	D	D	D	D	(T)
NEX	T	T	T	T	T	T	T	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X

PROB 8C (K,L) = (2,2)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	F	F	D	F	D	D	X
NAL	F	T	T	D	T	D	D	X
SOM	F	T	T	D	T	D	D	X
AON	D	D	D	D	D	D	D	X
NEX	F	T	T	D	T	D	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	F	F	D	F	D	D	X
NAL	F	T	T	D	T	D	D	X
SOM	F	T	T	D	T	D	D	X
AON	D	D	D	D	D	D	D	(F)
NEX	F	T	T	D	T	D	D	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	(F)	X	(F)	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	F	F	D	F	D	D	X
NAL	F	T	T	D	T	D	D	X
SOM	F	T	T	D	T	D	D	X
AON	D	D	D	D	D	D	D	(T)
NEX	F	T	T	D	T	D	D	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	(T)	X	(T)	(T)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	F	T	T	D	D	X
NAL	F	D	T	F	F	D	D	X
SOM	F	D	T	F	F	D	D	X
AON	D	D	D	D	D	D	D	X
NEX	F	D	T	F	F	D	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	F	T	T	D	D	X
NAL	F	D	T	F	F	D	D	X
SOM	F	D	T	F	F	D	D	X
AON	D	D	D	D	D	D	D	(F)
NEX	F	D	T	F	F	D	D	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	(F)	X	X	X	(F)	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	F	T	T	D	D	X
NAL	F	D	T	F	F	D	D	X
SOM	F	D	T	F	F	D	D	X
AON	D	D	D	D	D	D	D	(T)
NEX	F	D	T	F	F	D	D	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	(T)	X	X	X	(T)	(T)	X

PROB 8C (K,L) = (2,4)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	T	F	F	D	D	X
NAL	T	D	F	T	T	D	D	X
SOM	T	D	F	T	T	D	D	X
AON	D	D	D	D	D	D	D	X
NEX	T	D	F	T	T	D	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	T	F	F	D	D	X
NAL	T	D	F	T	T	D	D	X
SOM	T	D	F	T	T	D	D	X
AON	D	D	D	D	D	D	D	(F)
NEX	T	D	F	T	T	D	D	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	(F)	X	X	X	(F)	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	T	F	F	D	D	X
NAL	T	D	F	T	T	D	D	X
SOM	T	D	F	T	T	D	D	X
AON	D	D	D	D	D	D	D	(T)
NEX	T	D	F	T	T	D	D	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	(T)	X	X	X	(T)	(T)	X

PROB 8C (K,L) = (2,5)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X
NAL	F	D	F	D	T	F	D	X
SOM	F	D	F	D	T	F	D	X
AON	D	D	D	D	D	D	D	X
NEX	F	D	F	D	T	F	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X
NAL	F	D	F	D	T	F	D	X
SOM	F	D	F	D	T	F	D	X
AON	D	D	D	D	D	D	D	(F)
NEX	F	D	F	D	T	F	D	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	(F)	X	(F)	X	X	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X
NAL	F	D	F	D	T	F	D	X
SOM	F	D	F	D	T	F	D	X
AON	D	D	D	D	D	D	D	(T)
NEX	F	D	F	D	T	F	D	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	(T)	X	(T)	X	X	(T)	X

10

11

12

.DC.

MR-65

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	F	D	T	F	D	X
NAL	T	D	T	D	F	T	D	X
SOM	T	D	T	D	F	T	D	X
AON	D	D	D	D	D	D	D	X
NEX	T	D	T	D	F	T	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	F	D	T	F	D	X
NAL	T	D	T	D	F	T	D	X
SOM	T	D	T	D	F	T	D	X
AON	D	D	D	D	D	D	D	(F)
NEX	T	D	T	D	F	T	D	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	(F)	X	(F)	X	X	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	F	D	T	F	D	X
NAL	T	D	T	D	F	T	D	X
SOM	T	D	T	D	F	T	D	X
AON	D	D	D	D	D	D	D	(T)
NEX	T	D	T	D	F	T	D	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	(T)	X	(T)	X	X	(T)	X

13

PROB 8C (K,L) = (2,7)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	T	T	T	T	T	T	T	X
SOM	T	T	T	T	T	T	T	X
AON	D	D	D	D	D	D	D	X
NEX	T	T	T	T	T	T	T	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	T	T	T	T	T	T	T	X
SOM	T	T	T	T	T	T	T	X
AON	D	D	D	D	D	D	D	(F)
NEX	T	T	T	T	T	T	T	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	T	T	T	T	T	T	T	X
SOM	T	T	T	T	T	T	T	X
AON	D	D	D	D	D	D	D	(T)
NEX	T	T	T	T	T	T	T	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X

14

PROB 8C (K,L) = (2,8)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	F	F	F	F	F	F	F	X
SOM	F	F	F	F	F	F	F	X
AON	D	D	D	D	D	D	D	X
NEX	F	F	F	F	F	F	F	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	F	F	F	F	F	F	F	X
SOM	F	F	F	F	F	F	F	X
AON	D	D	D	D	D	D	D	(F)
NEX	F	F	F	F	F	F	F	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	F	F	F	F	F	F	F	X
SOM	F	F	F	F	F	F	F	X
AON	D	D	D	D	D	D	D	(T)
NEX	F	F	F	F	F	F	F	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X

15

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	F	T	T	D	D	X
NAL	D	D	D	D	D	D	D	X
SOM	F	D	T	F	F	D	D	X
AON	T	D	F	T	T	D	D	X
NEX	T	D	F	T	T	D	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	F	T	T	D	D	X
NAL	D	D	D	D	D	D	D	(F)
SOM	F	D	T	F	F	D	D	X
AON	T	D	F	T	T	D	D	X
NEX	T	D	F	T	T	D	D	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	(F)	X	X	X	(F)	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	F	T	T	D	D	X
NAL	D	D	D	D	D	D	D	(T)
SOM	F	D	T	F	F	D	D	X
AON	T	D	F	T	T	D	D	X
NEX	T	D	F	T	T	D	D	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	(T)	X	X	X	(T)	(T)	X

16

PROB 8C (K,L) = (3,4)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	T	F	F	D	D	X
NAL	D	D	D	D	D	D	D	X
SOM	T	D	F	T	T	D	D	X
AON	F	D	T	F	F	D	D	X
NEX	F	D	T	F	F	D	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	T	F	F	D	D	X
NAL	D	D	D	D	D	D	D	(F)
SOM	T	D	F	T	T	D	D	X
AON	F	D	T	F	F	D	D	X
NEX	F	D	T	F	F	D	D	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	(F)	X	X	X	(F)	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	T	F	F	D	D	X
NAL	D	D	D	D	D	D	D	(T)
SOM	T	D	F	T	T	D	D	X
AON	F	D	T	F	F	D	D	X
NEX	F	D	T	F	F	D	D	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	(T)	X	X	X	(T)	(T)	X

17

PROB 8C (K,L) = (3,5)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X
NAL	D	D	D	D	D	D	D	X
SOM	F	D	F	D	T	F	D	X
AON	T	D	T	D	F	T	D	X
NEX	T	D	T	D	F	T	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X
NAL	D	D	D	D	D	D	D	(F)
SOM	F	D	F	D	T	F	D	X
AON	T	D	T	D	F	T	D	X
NEX	T	D	T	D	F	T	D	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	(F)	X	(F)	X	X	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X
NAL	D	D	D	D	D	D	D	(T)
SOM	F	D	F	D	T	F	D	X
AON	T	D	T	D	F	T	D	X
NEX	T	D	T	D	F	T	D	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	(T)	X	(T)	X	X	(T)	X

18

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	F	D	T	F	D	X
NAL	D	D	D	D	D	D	D	X
SOM	T	D	T	D	F	T	D	X
AON	F	D	F	D	T	F	D	X
NEX	F	D	F	D	T	F	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	F	D	T	F	D	X
NAL	D	D	D	D	D	D	D	(F)
SOM	T	D	T	D	F	T	D	X
AON	F	D	F	D	T	F	D	X
NEX	F	D	F	D	T	F	D	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	(F)	X	(F)	X	X	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	F	D	T	F	D	X
NAL	D	D	D	D	D	D	D	(T)
SOM	T	D	T	D	F	T	D	X
AON	F	D	F	D	T	F	D	X
NEX	F	D	F	D	T	F	D	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	(T)	X	(T)	X	X	(T)	X

19

PROB 8C (K,L) = (3,7)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	D	D	D	D	D	D	D	X
SOM	T	T	T	T	T	T	T	X
AON	F	F	F	F	F	F	F	X
NEX	F	F	F	F	F	F	F	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	D	D	D	D	D	D	D	(F)
SOM	T	T	T	T	T	T	T	X
AON	F	F	F	F	F	F	F	X
NEX	F	F	F	F	F	F	F	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	D	D	D	D	D	D	D	(T)
SOM	T	T	T	T	T	T	T	X
AON	F	F	F	F	F	F	F	X
NEX	F	F	F	F	F	F	F	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X

20

PROB 8C (K,L) = (3,8)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	D	D	D	D	D	D	D	X
SOM	F	F	F	F	F	F	F	X
AON	T	T	T	T	T	T	T	X
NEX	T	T	T	T	T	T	T	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	D	D	D	D	D	D	D	(F)
SOM	F	F	F	F	F	F	F	X
AON	T	T	T	T	T	T	T	X
NEX	T	T	T	T	T	T	T	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	D	D	D	D	D	D	D	(T)
SOM	F	F	F	F	F	F	F	X
AON	T	T	T	T	T	T	T	X
NEX	T	T	T	T	T	T	T	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X

21

22

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMF
ALL	T	D	F	T	T	D	D	X
NAL	D	D	D	D	D	D	D	X
SOM	F	D	T	F	F	D	D	X
AON	T	D	F	T	T	D	D	X
NEX	T	D	F	T	T	D	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMF
ALL	T	D	F	T	T	D	D	X
NAL	D	D	D	D	D	D	D	F
SOM	F	D	T	F	F	D	D	X
AON	T	D	F	T	T	D	D	X
NEX	T	D	F	T	T	D	D	X
EXS	D	D	D	D	D	D	D	F
IND	D	D	D	D	D	D	D	F
IMP	X	F	X	X	X	F	F	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMF
ALL	T	D	F	T	T	D	D	X
NAL	D	D	D	D	D	D	D	T
SOM	F	D	T	F	F	D	D	X
AON	T	D	F	T	T	D	D	X
NEX	T	D	F	T	T	D	D	X
EXS	D	D	D	D	D	D	D	T
IND	D	D	D	D	D	D	D	T
IMP	X	T	X	X	X	T	T	X

PROB 8C (K,L) = (4,5)

23

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMF
ALL	F	D	F	D	T	F	D	X
NAL	D	D	D	D	D	D	D	X
SOM	T	D	T	D	F	T	D	X
AON	F	D	F	D	T	F	D	X
NEX	F	D	F	D	T	F	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMF
ALL	F	D	F	D	T	F	D	X
NAL	D	D	D	D	D	D	D	F
SOM	T	D	T	D	F	T	D	X
AON	F	D	F	D	T	F	D	X
NEX	F	D	F	D	T	F	D	X
EXS	D	D	D	D	D	D	D	F
IND	D	D	D	D	D	D	D	F
IMP	X	F	X	F	X	X	F	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMF
ALL	F	D	F	D	T	F	D	X
NAL	D	D	D	D	D	D	D	T
SOM	T	D	T	D	F	T	D	X
AON	F	D	F	D	T	F	D	X
NEX	F	D	F	D	T	F	D	X
EXS	D	D	D	D	D	D	D	T
IND	D	D	D	D	D	D	D	T
IMP	X	T	X	T	X	X	T	X

PROB 8C (K,L) = (4,6)

24

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMF
ALL	T	D	T	D	F	T	D	X
NAL	D	D	D	D	D	D	D	X
SOM	F	D	F	D	T	F	D	X
AON	T	D	T	D	F	T	D	X
NEX	T	D	T	D	F	T	D	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMF
ALL	T	D	T	D	F	T	D	X
NAL	D	D	D	D	D	D	D	F
SOM	F	D	F	D	T	F	D	X
AON	T	D	T	D	F	T	D	X
NEX	T	D	T	D	F	T	D	X
EXS	D	D	D	D	D	D	D	F
IND	D	D	D	D	D	D	D	F
IMP	X	F	X	F	X	X	F	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMF
ALL	T	D	T	D	F	T	D	X
NAL	D	D	D	D	D	D	D	T
SOM	F	D	F	D	T	F	D	X
AON	T	D	T	D	F	T	D	X
NEX	T	D	T	D	F	T	D	X
EXS	D	D	D	D	D	D	D	T
IND	D	D	D	D	D	D	D	T
IMP	X	T	X	T	X	X	T	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	D	D	D	D	D	D	D	X
SOM	F	F	F	F	F	F	F	X
AON	T	T	T	T	T	T	T	X
NEX	T	T	T	T	T	T	T	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	D	D	D	D	D	D	D	(F)
SOM	F	F	F	F	F	F	F	X
AON	T	T	T	T	T	T	T	X
NEX	T	T	T	T	T	T	T	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	D	D	D	D	D	D	D	(T)
SOM	F	F	F	F	F	F	F	X
AON	T	T	T	T	T	T	T	X
NEX	T	T	T	T	T	T	T	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X

25

PROB 8C (K,L) = (4,8)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	D	D	D	D	D	D	D	X
SOM	T	T	T	T	T	T	T	X
AON	F	F	F	F	F	F	F	X
NEX	F	F	F	F	F	F	F	X
EXS	D	D	D	D	D	D	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	D	D	D	D	D	D	D	(F)
SOM	T	T	T	T	T	T	T	X
AON	F	F	F	F	F	F	F	X
NEX	F	F	F	F	F	F	F	X
EXS	D	D	D	D	D	D	D	(F)
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	D	D	D	D	D	D	D	(T)
SOM	T	T	T	T	T	T	T	X
AON	F	F	F	F	F	F	F	X
NEX	F	F	F	F	F	F	F	X
EXS	D	D	D	D	D	D	D	(T)
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X

26

PROB 8C (K,L) = (5,5)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X
NAL	D	D	D	D	D	D	D	X
SOM	T	D	T	D	F	T	D	X
AON	D	D	D	D	D	D	D	X
NEX	F	D	F	D	T	F	D	X
EXS	T	D	T	D	F	T	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X
NAL	D	D	D	D	D	D	D	(F)
SOM	T	D	T	D	F	T	D	X
AON	D	D	D	D	D	D	D	(F)
NEX	F	D	F	D	T	F	D	X
EXS	T	D	T	D	F	T	D	X
IND	D	D	D	D	D	D	D	(F)
IMP	X	(F)	X	(F)	X	X	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X
NAL	D	D	D	D	D	D	D	(T)
SOM	T	D	T	D	F	T	D	X
AON	D	D	D	D	D	D	D	(T)
NEX	F	D	F	D	T	F	D	X
EXS	T	D	T	D	F	T	D	X
IND	D	D	D	D	D	D	D	(T)
IMP	X	(T)	X	(T)	X	X	(T)	X

27

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	F	D	T	F	D	X
NAL	D	D	D	D	D	D	D	X
SOM	F	D	F	D	T	F	D	X
AON	D	D	D	D	D	D	D	X
NEX	T	D	T	D	F	T	D	X
EXS	F	D	F	D	T	F	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	F	D	T	F	D	X
NAL	D	D	D	D	D	D	D	(F)
SOM	F	D	F	D	T	F	D	X
AON	D	D	D	D	D	D	D	(F)
NEX	T	D	T	D	F	T	D	X
EXS	F	D	F	D	T	F	D	X
IND	D	D	D	D	D	D	D	(F)
IMP	X	(F)	X	(F)	X	X	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	D	F	D	T	F	D	X
NAL	D	D	D	D	D	D	D	(T)
SOM	F	D	F	D	T	F	D	X
AON	D	D	D	D	D	D	D	(T)
NEX	T	D	T	D	F	T	D	X
EXS	F	D	F	D	T	F	D	X
IND	D	D	D	D	D	D	D	(T)
IMP	X	(T)	X	(T)	X	X	(T)	X

28

PROB 8C (K,L) = (5,7)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	D	D	D	D	D	D	D	X
SOM	F	F	F	F	F	F	F	X
AON	D	D	D	D	D	D	D	X
NEX	T	T	T	T	T	T	T	X
EXS	F	F	F	F	F	F	F	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	D	D	D	D	D	D	D	(F)
SOM	F	F	F	F	F	F	F	X
AON	D	D	D	D	D	D	D	(F)
NEX	T	T	T	T	T	T	T	X
EXS	F	F	F	F	F	F	F	X
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	D	D	D	D	D	D	D	(T)
SOM	F	F	F	F	F	F	F	X
AON	D	D	D	D	D	D	D	(T)
NEX	T	T	T	T	T	T	T	X
EXS	F	F	F	F	F	F	F	X
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X

29

PROB 8C (K,L) = (5,8)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	D	D	D	D	D	D	D	X
SOM	T	T	T	T	T	T	T	X
AON	D	D	D	D	D	D	D	X
NEX	F	F	F	F	F	F	F	X
EXS	T	T	T	T	T	T	T	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	D	D	D	D	D	D	D	(F)
SOM	T	T	T	T	T	T	T	X
AON	D	D	D	D	D	D	D	(F)
NEX	F	F	F	F	F	F	F	X
EXS	T	T	T	T	T	T	T	X
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	D	D	D	D	D	D	D	(T)
SOM	T	T	T	T	T	T	T	X
AON	D	D	D	D	D	D	D	(T)
NEX	F	F	F	F	F	F	F	X
EXS	T	T	T	T	T	T	T	X
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X

30

CO-ED

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X
NAL	D	D	D	D	D	D	D	X
SOM	T	D	T	D	F	T	D	X
AON	D	D	D	D	D	D	D	X
NEX	F	D	F	D	T	F	D	X
EXS	T	D	T	D	F	T	D	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X
NAL	D	D	D	D	D	D	D	(F)
SOM	T	D	T	D	F	T	D	X
AON	D	D	D	D	D	D	D	(F)
NEX	F	D	F	D	T	F	D	X
EXS	T	D	T	D	F	T	D	X
IND	D	D	D	D	D	D	D	(F)
IMP	X	(F)	X	(F)	X	X	(F)	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	D	T	D	F	T	D	X
NAL	D	D	D	D	D	D	D	(T)
SOM	T	D	T	D	F	T	D	X
AON	D	D	D	D	D	D	D	(T)
NEX	F	D	F	D	T	F	D	X
EXS	T	D	T	D	F	T	D	X
IND	D	D	D	D	D	D	D	(T)
IMP	X	(T)	X	(T)	X	X	(T)	X

PROB 8C (K,L) = (6,7)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	D	D	D	D	D	D	D	X
SOM	T	T	T	T	T	T	T	X
AON	D	D	D	D	D	D	D	X
NEX	F	F	F	F	F	F	F	X
EXS	T	T	T	T	T	T	T	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	D	D	D	D	D	D	D	(F)
SOM	T	T	T	T	T	T	T	X
AON	D	D	D	D	D	D	D	(F)
NEX	F	F	F	F	F	F	F	X
EXS	T	T	T	T	T	T	T	X
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X
NAL	D	D	D	D	D	D	D	(T)
SOM	T	T	T	T	T	T	T	X
AON	D	D	D	D	D	D	D	(T)
NEX	F	F	F	F	F	F	F	X
EXS	T	T	T	T	T	T	T	X
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X

PROB 8C (K,L) = (6,8)

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	D	D	D	D	D	D	D	X
SOM	F	F	F	F	F	F	F	X
AON	D	D	D	D	D	D	D	X
NEX	T	T	T	T	T	T	T	X
EXS	F	F	F	F	F	F	F	X
IND	D	D	D	D	D	D	D	X
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	D	D	D	D	D	D	D	(F)
SOM	F	F	F	F	F	F	F	X
AON	D	D	D	D	D	D	D	(F)
NEX	T	T	T	T	T	T	T	X
EXS	F	F	F	F	F	F	F	X
IND	D	D	D	D	D	D	D	(F)
IMP	X	X	X	X	X	X	X	X

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X
NAL	D	D	D	D	D	D	D	(T)
SOM	F	F	F	F	F	F	F	X
AON	D	D	D	D	D	D	D	(T)
NEX	T	T	T	T	T	T	T	X
EXS	F	F	F	F	F	F	F	X
IND	D	D	D	D	D	D	D	(T)
IMP	X	X	X	X	X	X	X	X

31

32

33

CO-MET

34

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
NAL	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
SOM	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
AON	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
NEX	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
EXS	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
IND	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
IMP	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

PROB 8C (K,L) = (7,8)

35

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X
NAL	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X
SOM	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X
AON	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X
NEX	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X
EXS	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X
IND	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X	F	F	F	F	F	F	F	X
IMP	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

PROB 8C (K,L) = (8,8)

36

	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP	ALL	NAL	SOM	AON	NEX	EXS	IND	IMP
ALL	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
NAL	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
SOM	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
AON	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
NEX	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
EXS	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
IND	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X	T	T	T	T	T	T	T	X
IMP	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

TT2 -- STOP

(b) Comments on Table 5 for QL-2 relation

In this case also, wherever differences occur, the second table in each row contains only F, while the third table contains only T, exactly as for SNS. Therefore, it is only necessary to work out the condition for the difference, if any, to arise. Just as for SNS, it is readily verified that

$$\underset{=}{t}(\underset{\sim}{a}\ ,\ \underset{\sim}{\underset{\sim}{q}}(\underset{\sim}{\underset{\sim}{k}})) = D \qquad (37a)$$

is the condition for the pair of values $(\underset{\sim}{a}\ ,\ \phi)$ to have ringed differences in the last column of Table 5, and similarly

$$\underset{=}{t}(\underset{\sim}{b}\ ,\ \underset{\sim}{\underset{\sim}{q}}(\underset{\sim}{\underset{\sim}{\ell}})) = D \qquad (37b)$$

is the condition for $(\phi\ ,\ \underset{\sim}{b})$ to lead to the ringed differences in the last row. These conditions are both necessary and sufficient, and are provable from the formulae for GLOGIC that have been given in MR-62. It is readily verified that they cover all examples in Table 6.

As has been indicated in the earlier parts I, II and III, the above differences can all be traced to one particular fact in GLOGIC, namely the difference between the concept of a null set/as a set which is impossible. Thus, in SNS logic as an empty set and

the state X = (0 0) means that neither T nor F is possible and is therefore a contradiction. On the other hand, in the GLOGIC of a two-element set, we can have (0 0) indicating merely the absence of both elements of the set, but no contradiction. In the same way, in QLOGIC, the vector (0 0 0) can represent the quantifier state ϕ meaning that the statement considered is neither true for all, nor for some, nor for none. On the other hand, the same vector (0 0 0) may be used to designate the absence of three elements a_1, a_2, a_3 of a three-element set. This distinction between absence of a set (zero set) and an impossible set (null set) has been noticed and considered in current applications of logic to the theory of sets, and it has, in fact, been incorporated in the computer language LISP. Here, in MATLOG, they can be incorporated, for all the logical relations involving the connectives A, O, I, E, by using the pure BVMF relations as GSTV and GUNPDT for the latter purpose, and the BVMF implementation of standard logical relations as GSTV2 and GUNPT2 for the former application. In one case, if the two sets represented by the vectors $\underset{\sim}{a}$ and $\underset{\sim}{b}$ are such that one of them is the zero set, while the other set

is "true", then the statement $\underset{\approx}{a} \vee \underset{\approx}{b}$ has the Boolean truth value 0 in pure BVMF, and 1 in the vector-matrix calculation <u>via</u> relative truth values of classical logic. The reason is simple — namely that if one of $\underline{a}$ or $\underline{b}$ is true, then it is immaterial if the other one is a zero set or not, and the disjunction $\underset{\approx}{a} \vee \underset{\approx}{b}$ is true in the logical formula. On the other hand, pure BVMF leads to its Boolean truth value (GBTV) to be 0 . As already mentioned in Part III, the latter is equivalent to the application of the so-called X-priority rule — namely that if either of the inputs is a <u>contradiction</u> (such as X or X X X), then the output of the conjunction is also X or X X X, i.e. a contradiction, irrespective of whatever be the truth value given by the relevant formula.

All the above considerations are true irrespective of whether we use them for SNS, QLOGIC or GLOGIC, and the general formula for GLOGIC can be applied with, or without, X-priority check, as required for the particular application for which it is needed. An indication of how this can be done in practice is given in the next section.

(c) Implementation of the X-priority rule

It is clear from what has been stated above, that for matrix relational operators, if the X-priority rule has to be included, then the subroutines GUNPDT, GBTV, GSTV are to be employed; while, if it should not be invoked, then the subroutines GUNPT2, GBTV2, GSTV2 should be employed, in GLOGIC. Corresponding formulae hold for SNS logic and quantifier logic (QL-2). This requirement has been already taken care of for PLOGIC (QL-1).

However, if the classical logic subroutines in BVMF — namely GUNPT2, GBTV2 and GSTV2 are systematically used in a program, and, in rare cases, the other three mentioned above are needed, they can be brought in by invoking the X-priority as follows:

First we define two null vectors, having MI and MJ zeros respectively, by

GVNUL1(I) = 0, I = 1, MI ; GVNUL2(I) = 0, I = 1, MJ (38a,b)

Then,

(i) For GUNPT2 (GVA,GMZ, MI, MJ) = GVB ,

if GVA = GVNUL1, then GVB = GVNUL2 (39a)

(ii) For GBTV2 (GVA, GMZ, GVB, MI, MJ) = BC,
if (GVA = GVNUL1) . OR . (GVB = GVNUL2),
then BC = 0 (39b

(iii) For GSTV2 (GVA, GMZ, GVB, MI, MJ) = SVC ,
if (GVA = GVNUL1) . OR . (GVB = GVNUL2) ,
then SVC = S4 (39c

It is to be emphasized that the X-priority rule does not hold for the pure Boolean operators $\underline{\underline{U}}$ and $\underline{\underline{V}}$ since they are invoked only in pure BVMF formulae and never in classical logic formulae (except in QL-1). Thus, for GUNION, Eq.(i) of page 2 of MR-62 is always true, and GUNION(GVA, GVNUL1, MI) GVA , and not GVNUL1

Ref. 1 : R. Kowalski, "Logic for Problem Solving",
North Holland, New York, 1979.

Boolean Vector-matrix Formulation (BVMF) of Logic

(Lecture Series 3)

G.N. Ramachandran

INSA Albert Einsetin Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

Matphil Reports No. 66, March 1987

Boolean Vector-matrix Formulation (BVMF) of Logic

(Lecture Series 3)

G.N. Ramachandran
INSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

Matphil Reports No. 66, March 1987

Boolean Vector-matrix Formulation (BVMF) of Logic

(Lecture Series 3)

Preface

This Matphil Report contains an account of three lectures dealing with propositional calculus and logical graphs in PC, implemented via BVMF. It is essentially a rehash of Lectures 1 and 2 of the Series-2, but contains several discussions of topics which are extensions of those lectures. Therefore, the report contains only a brief summary of Lecture-1, and the essentials of BA-2 algebra are given in Lecture-2. The essentially new aspects are contained in Lecture-3, where a de novo treatment of logical graphs is given. Even this is not quite upto-date at the moment of writing this preface because these lectures were delivered some six months ago, but they form a good introduction to the later application of these for obtaining computer programs for analyzing and implementing logical graphs. For instance, the procedures for reversal from contradictions and purifications could be made very much more compact in the attempt to make it into a computer program.

These notes have not been revised from the text as was prepared at the time of giving the lectures. They are useful for reference, but should be regarded as provisional, and as giving an introduction to our approach, rather than a definitive treatment of the theory.

Summary of Lecture-1 with the main points for discussion

1. Section 1 of Lecture-1 Series 2 was expanded. The association of classical logic of propositional calculus with BA-1 Boolean algebra is well-known. This associates the states T and F with 1 and 0, and the connectives $\vee$, $\wedge$, $\neg$ with $\oplus$, $\otimes$, ^{c}. The 2x2 truth tables for $\vee$ and $\wedge$ correspond to the $\otimes$ and $\oplus$ tables for $\otimes$ and $\ominus$.

2. The equations $a \vee b = c$ and $a \wedge b = c$ are called binary relations and require four equations corresponding to each a, b having the values T or F, and therefore is equivalent to the information contained in the truth tables of the two relations

3. From the truth table, we can also obtain a unary relation of the type $a \wedge (a \Rightarrow b) \Rightarrow b$. Thus, we shall ask the question,

What is b if a is false?

From the table of truth values given below

a \ b	T	F
T	1	0
F	1	1

it will be seen that

For $a = T$, $b = T$

For $a = F$, $b = T \vee F =$ tautology $= D$

In classical logic, we would say that a "false statement implies tautology" and $T \vee F$ = tautology.

Analogously we may ask,

What is b if $a \wedge (a \wedge b) \Longrightarrow b$

Here, using the truth table

a \ b	T	F
T	1	0
F	0	0

for "and", we obtain

For $a = T$, $b = T$

For $a = F$, $b = \neg T \wedge \neg F$ = contradiction = X

In classical logic, "$\neg a$ contradicts $a \wedge b$ ", or $T \wedge F$ = contradiction (Note —"inputs" are contradictory correspond to "output" is contradictory.)

4. This can be generalized to the question

If $a\ Z\ b = c = T$ and where $Z = \wedge, \vee, \Longrightarrow, \Longleftrightarrow, \not\Longleftrightarrow$,

what is the truth value of b given the truth value of a ?

We denote this by the unary relation $a\ Z = b$, and Table 1 summarizes the results. It will be seen that we need the new states, $D = T \vee F$ and $X = T \wedge F$, to describe the outputs of such unary relations.

Table 1. Unary relations in PC

a \ Z =	a Z = b for Z corresponding to				
	$\wedge$ = A	$\vee$ = O	$\Rightarrow$ = I	$\Leftrightarrow$ = E	$\not\Leftrightarrow$ = N
T	T	D	T	T	F
F	X	T	D	F	T

5. On extending the 2x2 truth tables for A = "and" O = "or" , we obtain Tables 2(a) and (c). However, these tables are not valid if a and b are the truth values of two data which refer to the same term, namely as $a_1 \wedge a_2 = a$ or $a_1 \vee a_2 = a$. In such a case, the proper truth tables are given in Table 2(b) and (d), for the connectives V(vidya) and U(union).

The inner 2x2 tables for V and U follow the rule that $a \wedge \neg a$ = contradiction and $a \vee \neg a$ = tautology. The extention of this to D and X follows standard logical principles.

6. It will be seen that these two tables 2(b) and 2(d) cannot be worked out according to the standard representation of "and" and "or" by the Boolean operators $\otimes$ and $\oplus$. We shall show that they are representable in BA-2 as two different operators, which we may symbolize by $\textcircled{\otimes}$ and $\textcircled{\oplus}$.

So also, two negation operators N and M have to be considered depending on the way that classical negation $\neg a = b$ is extended to D and X. As shown in Table 2(e), (f), these are denoted as $\neg a = b$ and $\overline{\neg} a = b$.

7. The idea of logical graphs was introduced, but this is reasonably well discussed in Section 3 of Lecture-1, Series 2 (p.14).

Table 2. Truth tables for Boolean and Matrix operators with T,F,D,X

$a \wedge b = c$ (a A b = c)

A	T	F	D	X
T	T	F	D	X
F	F	F	F	X
D	D	F	D	X
X	X	X	X	X

(a)

$a \otimes b = c$ (a V b = c)

V	T	F	D	X
T	T	X	T	X
F	X	F	F	X
D	T	F	D	X
X	X	X	X	X

(b)

$a \vee b = c$ (a O b = c)

O	T	F	D	X
T	T	T	T	X
F	T	F	D	X
D	T	D	D	X
X	X	X	X	X

(c)

$a \oplus b = c$ (a U b = c)

U	T	F	D	X
T	T	D	D	T
F	D	F	D	F
D	D	D	D	D
X	T	F	D	X

(d)

$(\neg a = b) \equiv$ (a N = b)

a	a N
T	F
F	T
D	D
X	X

(e)

$(\neg a = b) \equiv$ (a M = b)

a	a M
T	F
F	T
D	X
X	D

(f)

The essential new ideas are that any general logical argument (in PC, in particular) can be broken up into elementary statements of the types —

Binary forward : $a\ Z\ b = c \iff b\ Z^{(r)}\ a = c$

Binary reverse : $a(c, Z) = b \iff b(c, Z^{(r)}) = a$

Unary : $a\ Z = b \iff b\ Z^{(r)} = a$

The latter two are related to each other by

$$a(T, Z) = b \iff a\ Z = b$$

$$a(F, Z) = b \iff a\ Z^{c} = b$$

(Please see Lecture 2 — Series 2 —Section 2.)

These equations, valid for $Z = A$ and O in BA-1, will be explained.

For U, V, M we have only the relations given below:

Binary forward : $a\ B\ b = c$, $B = U$ or V

Unary : $a\ M = b$

6.(a) Distinction between $\wedge$ and $\otimes$: The distinctions between $\wedge$ and $\otimes$ indicated in Table 2 comes to the prominence when they are applied to the working out of compound sentences in propositional calculus. Two very simple examples are illustrated in Table 3 below. The description of these in terms of elementary statements is given below the table.

Table 3. Data illustrating the differences produced by parantheses and between $\wedge$ and $\otimes$ in logical statements

Sl No	Formula	Truth value c for		
		a = T, b = T	a = F, b = T	a = F, b = F
1	$a \wedge (\neg a \wedge b) = c$	$T \wedge F = F$	$F \wedge T = F$	$F \wedge F = F$
2	$(a \wedge \neg a) \wedge b = c$	$F \wedge T = F$	$F \wedge T = F$	$F \wedge F = F$
3	$a \otimes (\neg a \wedge b) = c$	$T \otimes F = X$	$F \otimes T = X$	$F \otimes F = F$
4	$(a \otimes \neg a) \wedge b = c$	$X \wedge T = X$	$X \wedge T = X$	$X \wedge F = X$
5	$a \wedge (\neg a \vee b) = c$	$T \wedge T = T$	$F \wedge T = F$	$F \wedge T = F$
6	$(a \wedge \neg a) \vee b = c$	$F \vee T = T$	$F \vee T = T$	$F \vee F = F$
7	$a \otimes (\neg a \vee b) = c$	$T \otimes T = T$	$F \otimes T = X$	$F \otimes T = X$
8	$(a \otimes \neg a) \vee b = c$	$X \vee T = X$	$X \vee T = X$	$X \vee F = X$

Examples of statements leading to Sl. Nos. 1 to 4

$a \Longleftrightarrow g$, $\neg a \wedge b \Longleftrightarrow h$, $g \wedge h \Longleftrightarrow c$ (1)

$a \Longleftrightarrow g$, $\neg a \Longleftrightarrow h$, $g \wedge b = c$ (2)

$a \Longleftrightarrow c$, $\neg a \wedge b \Longleftrightarrow c$ (3)

$a \Longleftrightarrow g$, $\neg a \Longleftrightarrow g$, $g \wedge b \Longleftrightarrow c$ (4)

Lecture - 2 of Series 3

1. Brief summary of Lecture 1

Essentially, we have shown that the BA-1 algebra, consisting of the Boolean numbers 1 and 0 and the operators $\oplus$, $\otimes$, ^{c}, can be associated with logical operators $\vee$, $\wedge$, $\neg$ of propositional calculus (PC); that the latter three have a double interpretation in the form of the BVMF connectives O, A, N and U, V, M. This becomes clear when the extended set of truth values T, F, D, X is considered and the logical operators $\vee$, $\wedge$, $\neg$ are extended to cover the new states D and X.

Although the 4x4 truth tables of all the operators are derivable from considerations of classical logic, their unequivocal representation in Boolean algebra cannot be done in BA-1.

A brief indication was made about the logical graph representation of unary and binary relations involving these operators. We observed that the "matrix connectives" O, A, N are relevant for the relation a Z b = c connecting information regarding truth values from two <u>different</u> sources for a and b, and that the relation involving "Boolean connectives" U, V, M,

are to be used when information regarding the same term a is obtained from two different sources a_1 and a_2 in the form a_1 B a_2 = a. This again is representable in very simple terms in BA-2.

2. Essentials of BA-2 representation

(Revision of Section 2 of Lecture 1 of the previous series)

States are represented by 2-vectors $(a_\alpha \quad a_\beta)$:

$$\underline{\underline{a}} = (a_\alpha \quad a_\beta) \ , \quad a_\alpha \ , \ a_\beta = 0, 1 \tag{2.1}$$

(The discussion in this section will be essentially limited to BA-2, and the general theory of BA-n, and the definition of matrix and Boolean operators for a general case, are reserved for the next lecture.)

The two generators of BA-2 are

$$T = s(1) = (1 \quad 0), \quad F = s(2) = (0 \quad 1) \tag{2.2a}$$

These are called "pure" states. Two "mixed" states are possible, denoted by

$$\begin{aligned} s(3) &= T \oplus F = (1 \quad 0) \oplus (0 \quad 1) = (1 \quad 1) = D \\ s(4) &= T \otimes F = (1 \quad 0) \otimes (0 \quad 1) = (0 \quad 0) = X \end{aligned} \tag{2.2b}$$

(a) Definition of "Boolean" connectives $\underline{\underline{V}}$, $\underline{\underline{U}}$, $\underline{\underline{M}}$

$$\underline{\underline{a}} \ \underline{\underline{V}} \ \underline{\underline{b}} \iff \underline{\underline{a}} \circledotimes \underline{\underline{b}} = \underline{\underline{c}} \iff a_\alpha \otimes b_\alpha = c_\alpha \ , \quad a_\beta \otimes b_\beta = c_\beta \tag{2.3}$$

(Boolean product of 2-vectors)

$$\underline{\underline{a}} \ \underline{\underline{U}} \ \underline{\underline{b}} \iff \underline{\underline{a}} \circledoplus \underline{\underline{b}} = \underline{\underline{c}} \iff a_\alpha \oplus b_\alpha = c_\alpha \ , \quad a_\beta \oplus b_\beta = c_\beta \tag{2.4}$$

(Boolean sum of 2-vectors)

We use $\circledotimes$ and $\circledoplus$ for these, to distinguish them from $\otimes$ and $\oplus$ in BA-1 algebra, in terms of which all expressions and formulae in BVMF are formulated.

In this representation, the double negation operator $\neg\neg = \underline{\underline{M}}$ is similarly the extension, from scalar to 2-vector form, of the complementation operator ^{c}. Thus,

$$\underline{\underline{a}}\,\underline{\underline{M}} = \underline{\underline{b}} \iff \neg\neg\underline{\underline{a}} = \underline{\underline{b}} \iff a_\alpha^c = b_\alpha\,;\quad a_\beta^c = b_\beta \tag{2.5}$$

It is readily verified that Tables 2(b, d and f) of Lecture 1 for the operators V, U, M are completely given by (2.3), (2.4) and (2.5) above. In other words, the properties of Boolean operators, which we introduced intuitively in Lecture 1, are completely representable by Boolean 2-vectors T, F, D, X. In particular, it is to be verified that the negation operator associated with this representation is $\underline{\underline{M}}$, and not $\underline{\underline{N}}$.

(b) Matrix operators

We shall give below the BA-2 representation for the three basic operators A, O, N of classical logic. Without making a formal derivation, we shall show that Tables 2(a, c, e) of Lecture 1 are fully representable by three matrix operators which we denote by the symbols $\wedge$, $\vee$, $\neg$ for $\underline{\underline{A}}$, $\underline{\underline{O}}$, $\underline{\underline{N}}$ as defined below.

$$\underline{\underline{A}}: \quad \underline{\underline{a}} \wedge \underline{\underline{b}} = \underline{\underline{c}} \iff a_\alpha \otimes b_\alpha = c_\alpha\,,\quad a_\beta \oplus b_\beta = c_\beta \tag{2.6}$$

$$\underline{\underline{O}}: \quad \underline{\underline{a}} \vee \underline{\underline{b}} = \underline{\underline{c}} \iff a_\alpha \oplus b_\alpha = c_\alpha\,,\quad a_\beta \otimes b_\beta = c_\beta \tag{2.7}$$

$$\underline{\underline{N}}: \quad \neg\underline{\underline{a}} = \underline{\underline{b}} \iff a_\beta = b_\alpha\,,\quad a_\alpha = b_\beta \tag{2.8}$$

We take these as empirically stated facts, but we shall show that it has a very elegant representation via 2x2 Boolean matrices, which are given in (2.9), (2.10) and (2.11). Denoting the matrix operators corresponding to A, O, N by $\underline{\underline{A}}$, $\underline{\underline{O}}$, $\underline{\underline{N}}$, and their 2x2 matrices by $|A|, |O|, |N|$, we have

$$\underline{\underline{a}}\,\underline{\underline{A}}\,\underline{\underline{b}} = \underline{\underline{c}} \iff \langle a|A|b\rangle = c_\alpha \;,\; \langle a|A^c|b\rangle = c_\beta \tag{2.9}$$

$$\underline{\underline{a}}\,\underline{\underline{O}}\,\underline{\underline{b}} = \underline{\underline{c}} \iff \langle a|O|b\rangle = c_\alpha \;,\; \langle a|O^c|b\rangle = c_\beta \tag{2.10}$$

$$\underline{\underline{a}}\,\underline{\underline{N}} = \underline{\underline{b}} \iff \langle a|N| = \langle b| \tag{2.11}$$

where

$$|A| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad |O| = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \quad |N| = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1.12a,b,c}$$

and the Dirac product notation for a general matrix connective $\underline{\underline{Z}}$ is given below :

$$\langle a|Z| = \langle b| : \bigoplus_\lambda a_\lambda \otimes Z_{\lambda\mu} = b_\mu \;;\; \lambda, \mu = \alpha, \beta \tag{2.13}$$

$$\langle a|Z|b\rangle = c \;:\; \bigoplus_\lambda \bigoplus_\mu a_\lambda \otimes Z_{\lambda\mu} \otimes b_\mu = c \;;\; \lambda, \mu = \alpha, \beta \tag{2.14}$$

and

$$Z^c_{\lambda\mu} = (Z_{\lambda\mu})^c \tag{2.15}$$

It will be noticed that the 2x2 matrices $|A|, |O|, |N|$ in (2.12) are identical with the 2x2 truth tables given in Lecture 1 for the matrix connective operators A, O, N . This is, in fact, to be taken / true for all matrix connectives in propositional calculus and this is the basis for our BVM formalism. Thus, it is assumed

that the Boolean 2x2 matrix in BA-2 representing an operator has, for its four elements, the same values (1 or 0) as in its truth table. This follows from the Dirac bracket formula (2.14) for the value of $\langle a|Z|b\rangle = c$. If the input $\underline{\underline{a}}$ and output $\underline{\underline{b}}$ are pure states, then only one a_λ and one b_μ will be present in the r.h.s of (2.14), and the truth value c of $\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}}$ is equal to $Z_{\lambda\mu}$, i.e. 1 or 0 corresponding to T and F of BA-1. Hence the construction of the truth table, as described in Lecture 1, will automatically lead to the identity between the truth table in BA-1, and the 2x2 matrix in BA-2, for the corresponding operator. As a consequence, everything requiring calculations using the information in the truth table can be worked out by matrix algebra. We shall illustrate the various types of such calculations below, and then give the summary of all these in the form of a few rules of derivation for BVMF, applied to PC.

(c) Matrix representation of various logical operations.

(i) Negation of terms and relations

The standard negation $\neg a$, $\neg b$ of $\underline{\underline{a}}$, $\underline{\underline{b}}$ is describable by the unary vector-matrix product given in (2.16a) below.

$$\underline{\underline{a}}\ \underline{\underline{N}} \leftrightarrow \langle a|N| = \langle a'| \ , \quad \underline{\underline{b}}\ \underline{\underline{N}} = \langle b|N| = \langle b'| \qquad (2.16a)$$

The validity of this is checked by the truth table 2(e) of Lecture 1 for the operator N in the relation $\underline{\underline{a}}\ \underline{\underline{N}} = \underline{\underline{b}}$.

Similarly, the negation of a relation $\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}} = \underline{\underline{c}}$ can be written in the form (2.16b)

$$\neg(\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}}) \iff \underline{\underline{a}}\ \underline{\underline{Z}}^{c}\ \underline{\underline{b}} = \neg\underline{\underline{c}} = \underline{\underline{c}}\ \underline{\underline{N}} \tag{2.16b}$$

where

$$(Z^{c})_{\lambda\mu} = (Z_{\lambda\mu})^{c} \tag{2.16c}$$

The justification of this follows from the fact that element $t^{c}_{\lambda\mu}$ in the truth table for the negated relation is the complement of $t_{\lambda\mu}$ for the non-negated relation.

Thus, the matrix for $a \Longrightarrow b$ is obtainable from the equation

$$\neg a \vee b \longleftrightarrow \underline{\underline{a}}\ \underline{\underline{N}}\ \underline{\underline{O}}\ \underline{\underline{b}} \iff \underline{\underline{a}}\ \underline{\underline{I}}\ \underline{\underline{b}} \longleftrightarrow a \Longrightarrow b \tag{2.17a}$$

Consequently, the matrix for "implication" is

$$|I| = |N|O| = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \tag{2.17b}$$

and it will be readily verified that the elements of $|I|$ are also the same as its truth table. Similarly, considering the negation of the implication relation, namely $\neg(a \Longrightarrow b) \iff a \wedge \neg b$, its matrix can be obtained as in (2.18a,b) :

$$\neg(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}) \iff \underline{\underline{a}} \wedge \neg\underline{\underline{b}} \longleftrightarrow \underline{\underline{a}}\ \underline{\underline{A}}\ \underline{\underline{N}}\ \underline{\underline{b}} \iff \underline{\underline{a}}\ \underline{\underline{I}}^{c}\ \underline{\underline{b}} \tag{2.18a}$$

Therefore, the matrix of $\underline{\underline{I}}^{c}$ is

$$|I^{c}| = |A|N| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = |I|^{c} \tag{2.18b}$$

The above examples illustrate the application of (2.16a) for the negation of a term, and (2.16b,c) for the negation of a relation, in BA-2 algebra. It also justifies the use of the notation $\underline{\underline{Z}}^{c}$ for the operator denoting the negation of the operator corresponding to the binary relation $\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}}$.

It will be noticed that the negation operator applied to the input $\underline{\underline{a}}$ is to the right of it, in the form $\underline{\underline{a}}\,\underline{\underline{N}}$, while the same operator, applied to the output $\underline{\underline{b}}$ of the relation, occurs to the left of it, as $\underline{\underline{N}}\,\underline{\underline{b}}$. This is in conformity with the standard practice adopted in conventional matrix algebra for a Dirac product of the type in Eq.(2.14). In our formulation, we follow the left-to-right sequence of the terms and operators, as employed in standard logic, and therefore all vectors have the standard form of the bra-vector $\langle a|$, $\langle b|$ etc., and the matrix is applied to the right of the vector. Therefore, in the triple product $\langle a|Z|b\rangle = c$, the second term will be represented by a ket-vector $|b\rangle$, and its negation, or the application of a matrix operator $\underline{\underline{N}}$ to it, will be represented by the matrix being to the left of it, in the form $|N\,|\,b\rangle$. This is the reason why in (2.17), the negation of $\underline{\underline{a}}$ is represented as $\underline{\underline{a}}\,\underline{\underline{N}}$, while in (2.18), the negation of $\underline{\underline{b}}$ is represented by $\underline{\underline{N}}\,\underline{\underline{b}}$

(ii) De Morgan relations in matrix notation

With the above formulae for representing the negation of terms and relations in propositional calculus in our BVMF notation, we shall give a proof of the two De Morgan relations via this formalism. Thus, the r.h.s of Eqs. (2.18a,b) are the

BVMF equivalents of the two De Morgan relations on the l.h.s.

$$\neg \underline{a} \wedge \neg \underline{b} = \neg(a \vee b) \Longleftrightarrow \underline{a}\,\underline{N}\,\underline{A}\,\underline{N}\,\underline{b} = \underline{a}\,\underline{O}^{c}\,\underline{b} \qquad (2.19a)$$

$$\neg \underline{a} \vee \neg \underline{b} = \neg(a \wedge b) \Longleftrightarrow \underline{a}\,\underline{N}\,\underline{O}\,\underline{N}\,\underline{b} = \underline{a}\,\underline{A}^{c}\,\underline{b} \qquad (2.19b)$$

In writing these, the rules for calculating $\neg a$ as $\langle a|N|$, and $\neg b$ as $|N|b\rangle$, are both employed, so that the matrix corresponding to the l.h.s is the "matrix product" of $|N|$, $|A|$, $|N|$ in succession in that sequence.

As can be readily verified, we get this matrix as

$$|N|A|N| = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} = \begin{pmatrix}0 & 0\\ 1 & 0\end{pmatrix}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} = \begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix} \qquad (2.20a)$$

$$= |O^{c}|\,, \text{ where } |O| = \begin{pmatrix}1 & 1\\ 1 & 0\end{pmatrix}, \text{ by (2.12b)} \qquad (2.20b)$$

This proves the first De Morgan relation. In an exactly similar manner, the second one can also be derived in the form

$$|N|O|N| = |A^{c}| \qquad (2.20c)$$

Both in the previous subsection (i), and in the description above, it will be noticed that the matrix product follows the same rules as for the product of two matrices in conventional algebra, with the difference that the multiplications and

additions involve only Boolean product and Boolean sum operations. Thus, the relevant formula is given in (2.21a,b):

$$|Z_1|\,Z_2| = |Z| \tag{2.21a}$$

is equivalent to the Boolean sum

$$\bigoplus_{\sigma}(Z_{1\lambda\sigma} \otimes Z_{2\sigma\lambda}) = Z_{\lambda\mu} \tag{2.21b}$$

(iii) Reversal of a relation expressed via a matrix connective

We shall illustrate this by proving the operator equation

$$\underline{\underline{b}}\ \underline{\underline{I}}\ \underline{\underline{a}} = \underline{\underline{a}}\ \underline{\underline{N}}\ \underline{\underline{I}}\ \underline{\underline{N}}\ \underline{\underline{b}} \tag{2.22a}$$

corresponding to the logical equation

$$(\underline{\underline{b}} \Longrightarrow \underline{\underline{a}}) \Longleftrightarrow (\neg a \Longrightarrow \neg b) \tag{2.22b}$$

For this, we employ the formula which is valid in a general Boolean algebra BA-n, namely that the matrix corresponding to the "reverse" of a relation $\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}}$ is represented by the relation $\underline{\underline{b}}\ \underline{\underline{Z}}^{(r)}\ \underline{\underline{a}}$, where the matrix for $Z^{(r)}$ is the transpose of the matrix for the forward relation. Thus,

$$|Z^{(r)}| = |Z^t| \tag{2.23a}$$

where the standard definition of the transpose of a matrix is carried over from conventional algebra, namely

$$Z^t_{\lambda\mu} = Z_{\mu\lambda} \tag{2.23b}$$

The proof of this, for a general m x n matrix, will be given later.

It will be readily seen that (2.26) and (2.28) both give the same matrix for $\underline{\underline{E}}$ which also agrees with its truth table. Thus, we arrive at a formula similar to (2.27), namely that the logical disjunction of two relations that co-exist between the terms a and b is representable by the Boolean sum of the corresponding matrices in the form (2.29), which can also be generalized to mxn matrices.

$$|Z_1| \oplus |Z_2| = |Z| \longleftrightarrow Z_{1\lambda\mu} \oplus Z_{2\lambda\mu} = Z_{\lambda\mu} \tag{2.29}$$

The justification of the Boolean product of the matrices for two conjoint relations, and the Boolean sum for the matrix of two disjoint relations, follow from the identity (2.30) in BA-1:

$$\left(\bigoplus_{\lambda} a_\lambda \otimes Z_{1\lambda\mu}\right) \underset{\otimes}{\oplus} \left(\bigoplus_{\lambda} a_\lambda \otimes Z_{2\lambda\mu}\right) = \bigoplus_{\lambda} a_\lambda \otimes (Z_{1\lambda\mu} \underset{\otimes}{\oplus} Z_{2\lambda\mu}),\ \lambda, \mu = \alpha, \beta \tag{2.30}$$

The matrix for the negation operator $\underline{\underline{N}}$ of (2.25b) which is the complement of the equivalence operator $\underline{\underline{E}}$, is obtainable from (2.16a) as :

$$|N| = |E^c| = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.31}$$

Thus, we verify the internal consistency of the association of the Boolean operations of $\oplus$, $\otimes$ and ^{c} of the corresponding elements of matrices for obtaining the disjunction, conjunction and negation of the two relations.

(d) Combination of one Boolean operator with another matrix operator

We have seen the essential formulae for Boolean connectives V and U, in BA-2, in Eqs. (2.3) and (2.4). We have also seen the various combinations of matrix operators in the above sections. In a similar manner, it is quite likely that combinations of the matrix operator N with the Boolean operator U or V can occur, as in

$$\neg \underline{\underline{a}} \otimes \neg \underline{\underline{b}} \Longleftrightarrow (\underline{\underline{a}}\ \underline{\underline{N}})\ \underline{\underline{V}}\ (\underline{\underline{N}}\ \underline{\underline{b}}) \qquad (2.32)$$

In such cases, the proper procedure would be to combine the matrix operators by the rules mentioned above, and the one to be given below for a string of matrix operators, and to apply the Boolean operator on the resultant terms. Thus, in the example given above, the calculation of $\neg \underline{\underline{a}} \otimes \neg \underline{\underline{b}}$ will be carried out in three steps as follows:

$$\underline{\underline{a}}\ \underline{\underline{N}} = \underline{\underline{a}}',\quad \underline{\underline{b}}\ \underline{\underline{N}} = \underline{\underline{b}}',\quad \underline{\underline{a}}'\ \underline{\underline{V}}\ \underline{\underline{b}}' = \underline{\underline{c}} \Longleftrightarrow \underline{\underline{a}}\ \underline{\underline{N}}\ \underline{\underline{V}}\ \underline{\underline{N}}\ \underline{\underline{b}} = \underline{\underline{c}} \qquad (2.33)$$

However, if a string of Boolean relations containing either V or U occurs, it can be implemented as in (2.34a,b) :

$$\underline{\underline{a}} \otimes \underline{\underline{b}} \otimes \ldots \otimes \underline{\underline{f}} = \underline{\underline{x}} \Longleftrightarrow a_\alpha \otimes b_\alpha \otimes \ldots \otimes f_\alpha = x_\alpha\ ,\quad a_\beta \otimes b_\beta \otimes \ldots \otimes f_\beta = x_\beta \qquad (2.34a)$$

$$\underline{\underline{a}} \oplus \underline{\underline{b}} \oplus \ldots \oplus \underline{\underline{f}} = \underline{\underline{x}} \Longleftrightarrow a_\alpha \oplus b_\alpha \oplus \ldots \oplus f_\alpha = x_\alpha\ ;\quad a_\beta \oplus b_\beta \oplus \ldots \oplus f_\beta = x_\beta \qquad (2.34b)$$

In (2.34a,b) the individual terms $\underline{\underline{a}}, \underline{\underline{b}}, \ldots \underline{\underline{f}}$ can themselves be expressions involving matrix connectives which can be implemented by the formulae discussed for these connectives.

(e) Complete list of 2x2 matrix connectives

As we have seen above, only three types of matrix connectives have to be considered, namely disjunction ($\underline{\underline{O}}$), conjunction ($\underline{\underline{A}}$) and equivalence $\underline{\underline{E}}(\underline{\underline{N}})$. In the former two cases, there are four matrices of each type obtained by having negation, or affirmation, for $\underline{\underline{a}}$ and $\underline{\underline{b}}$, in the relation $\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}}$. We shall employ the notation $\underline{\underline{Z}}(k, \ell)$ with $k, \ell =$ 1 or 2 according as the BA-2 state of $\underline{\underline{a}}$ and $\underline{\underline{b}}$ is s(1) or s(2). Then, we obtain four $\underline{\underline{O}}(k, \ell)$, four $\underline{\underline{A}}(k, \ell)$ and two different $\underline{\underline{E}}(k, \ell)$, namely $\underline{\underline{E}} = \underline{\underline{E}}(1, 1) = \underline{\underline{E}}(2, 2)$ and $\underline{\underline{N}} = \underline{\underline{E}}(1, 2) = \underline{\underline{E}}(2, 1)$. These ten matrices are listed in Table 1. The four possible connectives $\underline{\underline{I}}(k, \ell)$ and $\underline{\underline{J}}(k, \ell)$, for implication and reverse implication, are not distinct from $\underline{\underline{O}}(k, \ell)$ and they can be obtained from the latter by the equations $\underline{\underline{I}}(k, \ell) = \underline{\underline{O}}(k^c, \ell)$, $\underline{\underline{J}}(k, \ell) = \underline{\underline{O}}(k, \ell^c)$, where $k^c, \ell^c = 2, 1$ according as $k, \ell = 1, 2$.

As we have seen, all possible matrix connectives are representable by 2x2 Boolean matrices which contain four Boolean numbers, 0 or 1. Consequently, there are only 16 possible matrices in BA-2 of which we have exhausted ten in listing the ten different connectives in PC. For completeness, we list the

Table 1. The ten connectives of propositional calculus and their BA-2 matrices

$\underline{A}$(and)	$\underline{N}\ \underline{A}$ (not implicate J^c)	$\underline{A}\ \underline{N}$ (not imply $\underline{I}^c$)	$\underline{N}\ \underline{A}\ \underline{N}$ (nor $\underline{O}^c$)
$\underline{a} \wedge \underline{b} = \underline{c}$	$\neg \underline{a} \wedge \underline{b} = \underline{c}$	$\underline{a} \wedge \neg \underline{b} = \underline{c}$	$\neg \underline{a} \wedge \neg \underline{b} = \underline{c}$
$\underline{a}\ \underline{A}\ \underline{b} = \underline{c}$	$\underline{a}\ \underline{N}\ \underline{A}\ \underline{b} = \underline{c}$	$\underline{a}\ \underline{A}\ \underline{N}\ \underline{b} = \underline{c}$	$\underline{a}\ \underline{N}\ \underline{A}\ \underline{N}\ \underline{b} = \underline{c}$
$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \underline{A}(1,1)$	$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \underline{A}(2,1)$	$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = \underline{A}(1,2)$	$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \underline{A}(2,2)$
$\underline{O}$ (or)	$\underline{N}\ \underline{O}$(imply = $\underline{I}$)	$\underline{O}\ \underline{N}$ (reverse* imply $\underline{J}$)	$\underline{N}\ \underline{O}\ \underline{N}$ (nand, $\underline{A}^c$)
$\underline{a} \vee \underline{b} = \underline{c}$	$\neg \underline{a} \vee \underline{b} = \underline{c}$	$\underline{a} \vee \neg \underline{b} = \underline{c}$	$\neg \underline{a} \vee \neg \underline{b} = \underline{c}$
$\underline{a}\ \underline{O}\ \underline{b} = \underline{c}$	$\underline{a}\ \underline{N}\ \underline{O}\ \underline{b} = \underline{c}$	$\underline{a}\ \underline{O}\ \underline{N}\ \underline{b} = \underline{c}$	$\underline{a}\ \underline{N}\ \underline{O}\ \underline{N}\ \underline{b} = \underline{c}$
$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \underline{O}(1,1)$	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \underline{O}(2,1)$	$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \underline{O}(1,2)$	$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \underline{O}(2,2)$
$\underline{E}$ (Equivalent)	$\underline{N}$(Negation) = $\underline{E}^c$	$\underline{I} = \underline{N}\ \underline{O}$ (imply) (suff)	
$(\underline{a} \equiv \underline{b}) = \underline{c}$	$(\underline{a} \not\equiv \underline{b}) = \underline{c}$	$\underline{J} = \underline{O}\ \underline{N}$ (reverse imply*) (nec.)	
$\underline{a}\ \underline{E}\ \underline{b} = \underline{c}$	$\underline{a}\ \underline{N}\ \underline{b} = \underline{c}$	$\underline{I}^t = \underline{J} = \underline{N}\ \underline{I}\ \underline{N}$ (contrapositive form)	
$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \underline{E}(1,1)$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \underline{E}(2,1)$	$(\underline{b} \Rightarrow \underline{a}) \equiv (\neg \underline{a}) \Rightarrow (\neg \underline{b})$	

*The name "reverse imply" stands for the relation $\underline{a} \Leftarrow \underline{b}$

$$|Z(k,\ell)| = |Z(1,1)|, |N|Z(1,1)|, |Z(1,1)|N|, |N|Z(1,1)|N|,$$

$$\text{for } (k,\ell) = (1,1), (1,2), (2,1), (2,2) \text{ respectively} \qquad (2.35)$$

Therefore, all the procedures described above in this section can be implemented in terms of matrix $\underline{\underline{Z}}(k,\ell)$. The relevant formulae for the unary and binary form of implementation of the relation $\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}}$ is as follows:

Binary form

$$\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}} = \underline{\underline{c}} \iff \langle a|Z|b\rangle = c_\alpha\ ,\ \langle a|Z^c|b\rangle = c_\beta\ ;\ \langle c| = (c_\alpha\ \ c_\beta) \qquad (2.36)$$

Unary form

$$\underline{\underline{a}}\ \underline{\underline{Z}} = \underline{\underline{b}} \iff \langle a|Z| = \langle b|\ ,\ |Z| = |Z(k,\ell)| \qquad (2.37)$$

For a sequence of unary or binary matrix relations, the following equations (2.38) and (2.39) can be employed.

$$\underline{\underline{a}}\ \underline{\underline{Z}}_1\ \underline{\underline{b}}\ \underline{\underline{Z}}_2\ \underline{\underline{c}}\ \cdots\ \underline{\underline{e}}\ \underline{\underline{Z}}_n\ \underline{\underline{f}} = \underline{\underline{x}} \iff \langle a|Z_1|b\rangle = \langle b'|,\ \langle b'|Z_2|c\rangle = \langle c'|$$

$$,\ldots,\ \langle e'|Z_n|f\rangle = \langle x| \qquad (2.38)$$

and

$$\underline{\underline{a}}\ \underline{\underline{Z}}_1\ \underline{\underline{Z}}_2\ \cdots\ \underline{\underline{Z}}_n = \underline{\underline{y}} \iff \langle a|Z| = \langle y|, \text{where } |Z| = |Z_1|Z_2|\ \ldots|Z_n|$$

$$(2.39)$$

We shall give below a resumé of Section 2 for elementary relations and their implementation. In a general logical graph,

several of these will be combined in various combinations and sequences. The general procedures for treating such logical graphs are discussed in Sections 3 and 4, and considerations connected with the analysis and the implementation of a logical argument in PC using such logical graphs will be discussed in Lecture 3.

Section 3 will practically follow the treatment in Section 3 of Lecture 1, Series 2.

(g) Summary of Section 2

In BA-2 representation of PC we can employ two-vectors and four truth values — two of them, (1 0) and (0 1), for the pure states T and F, and two others, (1 1) and (0 0), for the mixed states D and X. Of these, only three, namely T, F, D are permissible states, while the fourth (X) indicates contradiction when it is obtained as an output of a relation, and will not be used as an input of any step.

The implementation of an argument in PC proceeds via single steps, in which an elementary relation is implemented at a time, and this can be either in the unary form, or in the binary forward, or binary reverse, form, as mentioned in Lecture 1. These steps

make use of a matrix connective $\underline{\underline{Z}}$ (= $\underline{\underline{A}}(k, \ell)$, $\underline{\underline{O}}(k, \ell)$, $\underline{\underline{I}}(k, \ell)$ $\underline{\underline{E}}$ or $\underline{\underline{N}}$) if two independent inputs $\underline{\underline{a}}$ and $\underline{\underline{b}}$ are employed in a binary forward relation to yield the output $\underline{\underline{c}}$, or when the binary reverse relation $\underline{\underline{a}}\ (\underline{\underline{c}}, \underline{\underline{Z}}) = \underline{\underline{b}} \Longleftrightarrow \underline{\underline{a}}\ \underline{\underline{Z}}' = \underline{\underline{b}}$, is employed with $\underline{\underline{a}}$ and $\underline{\underline{b}}$ being two different terms. However, if concurrent information in the form $\underline{\underline{a}}_1$ and $\underline{\underline{a}}_2$ have to be combined, by conjunction, disjunction, or complementation, to yield the truth value of $\underline{\underline{a}}$, then the connective to be employed is the Boolean operator $\underline{\underline{B}}$ (= $\underline{\underline{U}}$, $\underline{\underline{V}}$ or $\underline{\underline{M}}$). More generally, whenever two entities (terms or relations) refer to the same output and they are combined, then the Boolean operation of $\underline{\underline{U}}$, $\underline{\underline{V}}$, or $\underline{\underline{M}}$, is applied to the output terms, or to the relations, as the case may be, while if they refer to different inputs, leading to one output, then the matrix connectives $\underline{\underline{A}}$, $\underline{\underline{O}}$ and $\underline{\underline{N}}$ and their combinations are employed.

Several operations such as negation, reversal of unary relation, compounding of relations in parallel, or in sequence, can all be manipulated via BA-1 operations on the elements of the matrices and vectors employed in BA-2. A short list of these are as follows:

Two matrix operators in succession — The Boolean algebraic equation for the matrix $|Z|$ corresponding to $\underline{\underline{Z}} = \underline{\underline{Z}}_1\ \underline{\underline{Z}}_2$ is $|Z_1|\ Z_2| = |Z|$ where $\bigoplus_{\sigma} Z_{1\lambda\sigma} \otimes Z_{2\sigma\mu} = Z_{\lambda\mu}$.

Two Boolean operators in succession : $\underline{a}\ \underline{B}_1\ \underline{b} = \underline{b}'$, $\underline{b}'\ \underline{B}_2\ \underline{c} = \underline{d}$, are equivalent to $\underline{a}\ \underline{B}_1\ \underline{b}\ \underline{B}_2\ \underline{c} = \underline{d}$ can be implemented, based on (2.34a,b), in the form

$$a_\alpha\ B_1\ b_\alpha\ B_2\ c_\alpha = d_\alpha\ ,\quad a_\beta\ B_1\ b_\beta\ B_2\ c_\beta = d_\beta$$

where B_1, B_2 can be U or V. The following particular cases of matrix operators may be mentioned.

Negation of a term is represented by the matrix operator $\underline{N}$, as in $\neg\underline{a} \vee \underline{b} \iff (\underline{a}\ \underline{N})\ \underline{O}\ (\underline{b}) \iff \underline{a}\ \underline{Z}\ \underline{b}$, where $\underline{Z} = \underline{N}\ \underline{O}$.

Negation of a relation is expressed by the complementation of the elements of the corresponding matrix, as in $Z^c_{\lambda\mu} = (Z_{\lambda\mu})^c$, as in $\underline{a}\ \underline{N}\ \underline{A}\ \underline{N}\ \underline{b} = \underline{a}\ \underline{O}^c\ \underline{b}$.

Reversal of a unary relation is expressed by $\underline{Z}^{(r)} = \underline{Z}^{(t)}$, with $|Z^{(r)}|$ being the transpose of the matrix $|Z|$ for the forward relation, which is given by $Z^{(r)}_{\lambda\mu} = Z_{\mu\lambda}$.

Compounding of relations are of three types

(a) Two relations in parallel : These are again of two types.

Disjoint : $\underline{a}\ \underline{Z}_1\ \underline{b} \vee \underline{a}\ \underline{Z}_2\ \underline{b} = \underline{a}\ \underline{Z}\ \underline{b}$ where $|Z| = |Z_1| \oplus |Z_2|$

Conjoint : $\underline{a}\ \underline{Z}_1\ \underline{b} \wedge \underline{a}\ \underline{Z}_2\ \underline{b} = \underline{a}\ \underline{Z}\ \underline{b}$ where $|Z| = |Z_1| \otimes |Z_2|$

with $Z_{\lambda\mu} = Z_{1\lambda\mu} \oplus Z_{2\lambda\mu}$ and $Z_{1\lambda\mu} \otimes Z_{2\lambda\mu}$ in the two cases

(b) Two relations in series

Unary relation : $\underline{a}\ \underline{Z}_1 = \underline{b}$, $\underline{b}\ \underline{Z}_2 = \underline{c} \iff \underline{a}\ \underline{Z} = \underline{c}$

with $\underline{Z}_1 . \underline{Z}_2 = \underline{Z}$ where $|Z_1|\,|Z_2| = |Z|$.

Binary relation: $\underline{a}\ \underline{Z}_1\ \underline{b}\ \underline{Z}_2\ \underline{c} = \underline{d} \iff \underline{a}\ \underline{Z}_1\ \underline{b} = \underline{b}'$, $\underline{b}'\ \underline{Z}_2\ \underline{c} = \underline{d}$

It is believed that the implementation of a general argument or proof in propositional calculus, including back-tracking from a contradiction, can be performed by utilizing one or more of the above processes.

Series-3
Lecture-3
18-3-87

Logical graphs and their implementation

(Revision of Lecture 2,Series 2)

This lecture will deal with the construction of a logical graph for a general argument in propositional calculus, theorems relating to their implementation, and algorithms connected with the practical application of these ideas. Suitable examples are given for illustrating the procedures described here.

The first part of the lecture will lead up to a theorem showing that any logical graph, representing a set of relations in propositional calculus, can be implemented in a sequential manner, by breaking down the argument into elementary relations and rearranging them suitably. In the second part of the lecture, two examples will be given of this procedure and then an algorithm for this purpose will be worked out, for the forward implementation of the graph. Similarly, in the particular examp the reversal for back-tracking from a contradiction to its source will also be illustrated, and later, a general algorithmic scheme for this back-tracking will be worked out.

Series-3
Lecture-3
18.3.87

1. Elements of logical graphs

(a) Matrix connectives

Binary relation

Basic form of binary relation is

$$\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}} = \underline{\underline{c}} \quad , \text{ standing for } \underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}} \Longleftrightarrow \underline{\underline{c}} \tag{3.1a}$$

in standard logic. This is applied for

$$\underline{\underline{Z}} = \underline{\underline{A}}(k, \ell),\ \underline{\underline{O}}(k, \ell),\ \underline{\underline{E}}(k, \ell),\ \underline{\underline{I}}(k, \ell) \text{ as } \underline{\underline{O}}(k^n, \ell) \tag{3.1b}$$

Binary reverse

(equivalent to unary relation of Lecture-1 (Series 2), page 14)

As indicated in Fig.1, this may be implemented either in the binary reverse form, or the unary form. In classical logic, the statement has the forms $\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}}$, $\neg(\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}})$. This corresponds to the first two possibilities in (3.2a)

$$\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}} = T, F, D \Longleftrightarrow \underline{\underline{a}}\ \underline{\underline{Z}} = \underline{\underline{b}},\ \underline{\underline{a}}\ \underline{\underline{Z}}^c = \underline{\underline{b}},\ \underline{\underline{a}}\ \underline{\underline{D}} = \underline{\underline{b}} \tag{3.2a}$$

This is implemented either in the binary reverse form, or the unary form, according to the problem, as follows:

If $\underline{\underline{Z}}$ or $\underline{\underline{Z}}^c$ in the r.h.s of (3.2a) is of the $\underline{\underline{I}}(\underline{\underline{O}})$-type or D-type, or $\underline{\underline{E}}$-type, we implement the unary relation

$$\underline{\underline{a}}\ \underline{\underline{Z}}' = \underline{\underline{b}} \quad , \quad \underline{\underline{Z}}' = \underline{\underline{Z}} \text{ or } \underline{\underline{Z}}^c \tag{3.2b}$$

If $\underline{\underline{Z}}$ or $\underline{\underline{Z}}^c$/is of the A-type, we implement it in the binary for $\underline{\underline{Z}}'$ reverse form, namely

$$\underline{\underline{a}}\ \underline{\underline{Z}}'\ \underline{\underline{b}} = \underline{\underline{c}}', \quad \underline{\underline{c}}'' = T, F, D, \quad \underline{\underline{c}}'\ \underline{\underline{V}}\ \underline{\underline{c}}'' = \underline{\underline{c}} \tag{3.2c}$$

In all the above types of implementation of logical elementary relations involving matrix connectives, the contradictory state can occur only for (3.2c) from the vidya check. If the output $\underline{\underline{c}}$ of this check is $\mathbf{X}$, the processing of the argument is stopped. If not, the "revised output" $\underline{\underline{c}}$ is used thereafter for further implementation of the graph.

(b) Boolean connectives

This can occur only as binary relations, and of these $\underline{\underline{U}}$ is to be used only under rare conditions. On the other hand, the vidya operator $\underline{\underline{V}}$ is used for consistency check between multiple inputs to a given term $\underline{\underline{a}}$ from two sources as $\underline{\underline{a}}'$ and $\underline{\underline{a}}''$. Then

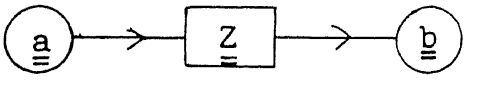
$$\underline{\underline{a}}'\ \underline{\underline{V}}\ \underline{\underline{a}}'' = \underline{\underline{a}} \tag{3.3}$$

Whenever there are more than one input to a term, the vidya check is applied successively between pairs of them as mentioned in Section 2.

As with (3.2c), the vidya check in (3.3) can also lead to contradictions, when the implementation is stopped and the source of the contradiction searched for, by back-tracking.

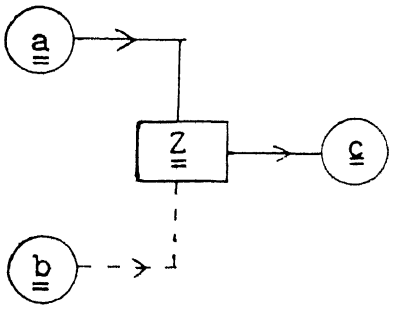
(c) Elementary building blocks of all logical graphs

A logical argument is built up of elementary blocks of the three types mentioned above, namely matrix-connective binary relation, matrix unary or binary reverse relation and Boolean connective binary relation. Therefore, in studying the structure of the logical graph representing a general argument, we need consider only the different ways in which the two basic structures, unary and binary relation, can be combined. These are indicated in Fig.1



(Unary relation)

$\underline{\underline{a}}\ \underline{\underline{Z}} = \underline{\underline{b}}$



$\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}} = \underline{\underline{c}}$

(Binary relati

Fig.1 The only two types of elementary sub-graphs which form the building blocks of a logical graph (in propositional calculus). In this $\underline{\underline{Z}}$ may be a matrix or Boolean connective.

2. Forward implementation of a logical graph

(a) Representation of special features

In the last subsection 1(c), we indicated that only two standard building blocks, namely a unary relation and a binary relation, implemented in the forward direction, are needed for constructing a logical graph. In this subsection, we shall show how more complicated features can be implemented in terms of these, before indicating the construction of the full graph.

(i) Binary reverse: This has the form $\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}} = T, F$ which is implemented in the form

$$\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}} = \underline{\underline{c}}' \ , \ T, F = \underline{\underline{c}}'' \ , \ \underline{\underline{c}}'\ \underline{\underline{V}}\ \underline{\underline{c}}'' = \underline{\underline{c}} \qquad (3.4)$$

as already indicated in (3.2). The block diagram for this is given in Fig. 2(a), and as will be seen from it, it consists of two binary blocks in sequence.

(ii) Multiple inputs to a term of the type $\underline{\underline{a}}_1\ \underline{\underline{V}}\ \underline{\underline{a}}_2\ \underline{\underline{V}}\ \underline{\underline{a}}_3 = \underline{\underline{a}}$

This again is implemented as two successive binary relations of the type

$$\underline{\underline{a}}_1\ \underline{\underline{V}}\ \underline{\underline{a}}_2 = \underline{\underline{a}}_2' \ , \quad \underline{\underline{a}}_2'\ \underline{\underline{V}}\ \underline{\underline{a}}_3 = \underline{\underline{a}} \qquad (3.5)$$

Its block diagram is given in Fig. 2(b). This can obviously be extended to any number of inputs to a term $\underline{\underline{a}}$.

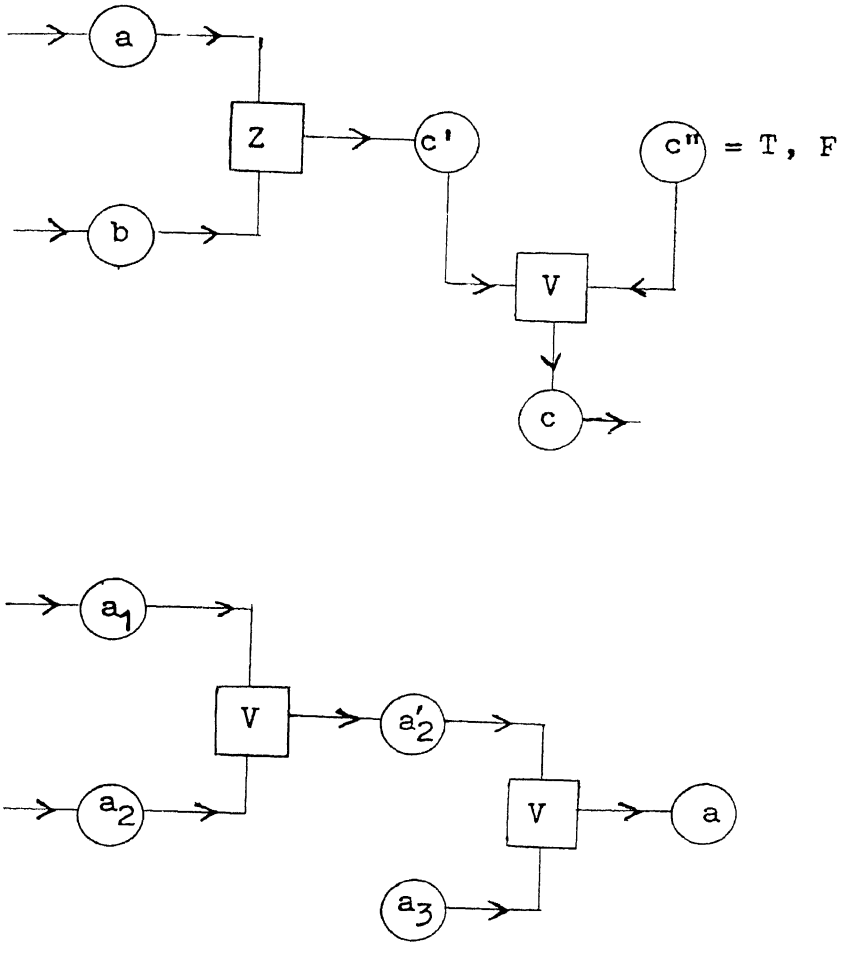


Fig.2. The construction of the vidya checks contained in (a) binary reverse relation and (b) multiple inputs to a term .

(iii) Multiple outputs from a connective of the type

$\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}} = \underline{\underline{c}}_1, \underline{\underline{c}}_2, \underline{\underline{c}}_3$

This example will be implemented as

$$\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}} = \underline{\underline{c}}_1\ ,\quad \underline{\underline{c}}_1\ \underline{\underline{E}} = \underline{\underline{c}}_2\ ,\quad \underline{\underline{c}}_2\ \underline{\underline{E}} = \underline{\underline{c}}_3 \tag{3.6}$$

Its block diagram is given in Fig. 3(a).

(iv) The same term serving as inputs to a number of relations

An example of this is

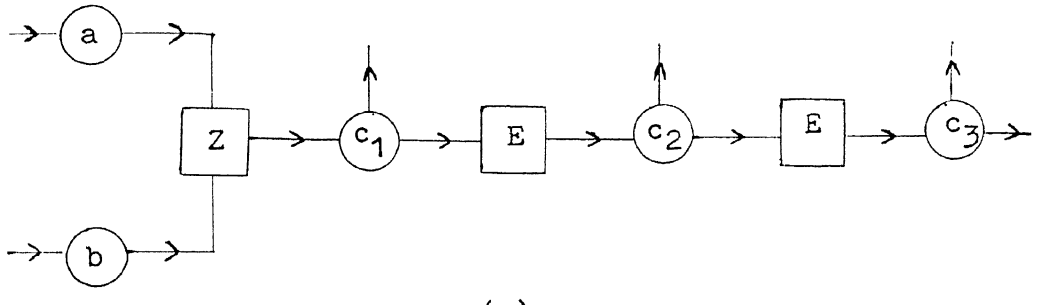
$$\underline{\underline{a}}\ \underline{\underline{Z}}_1 = \underline{\underline{b}}\ ,\quad \underline{\underline{a}}\ \underline{\underline{Z}}_2\ \underline{\underline{c}} = \underline{\underline{d}}\ ,\quad \underline{\underline{a}}\ \underline{\underline{Z}}_3 = \underline{\underline{e}} \tag{3.7}$$

This is implemented as follows:

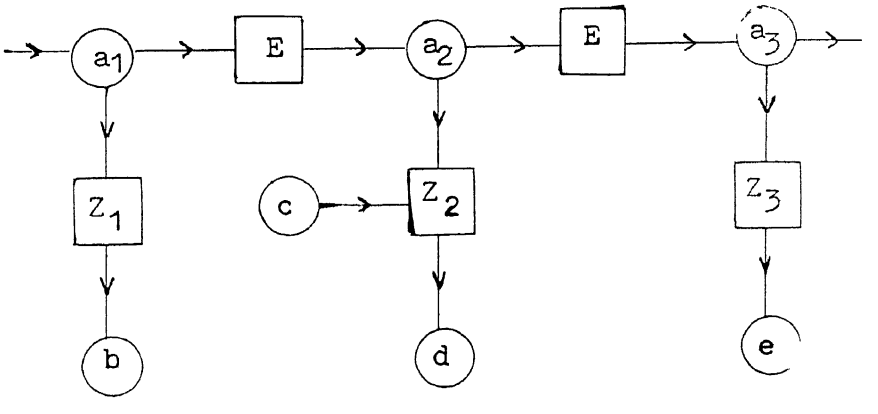
$$\underline{\underline{a}}_1\ \underline{\underline{Z}}_1 = \underline{\underline{b}}\ ,\quad \underline{\underline{a}}_1\ \underline{\underline{E}} = \underline{\underline{a}}_2\ ,\quad \underline{\underline{a}}_2\ \underline{\underline{Z}}\ \underline{\underline{c}} = \underline{\underline{d}}\ ,\quad \underline{\underline{a}}_2\ \underline{\underline{E}} = \underline{\underline{a}}_3,\quad \underline{\underline{a}}_3\ \underline{\underline{Z}}_3 = \underline{\underline{e}} \tag{3.8}$$

and its block diagram is shown in Fig. 3(b).

With these modifications, a general graph will have the property that one, or at most two, edges lead into a connective, according as it represents a unry relation or binary relation, and only one edge leads out of a connective. Similarly, only one edge leads to any term from a connective, and at most two edges lead out of a term. Under these conditions, it can be shown that, if there are no directed circuits as mentioned below, then a logical graph has at least one input and one output term, which is required for the theorem of sequential implementability to be proved below.



(a)



(b)

Fig.3. Method of treating multiple edges from (a) a connective to a number of terms, (b) a term to a number of connectives , so as to simplify the theoretical interpretation of a logical graph.

The constructions in Fig. 2(a) and (b) are to be incorporated in every graph before it is practically implemented. Those in Fig. 3(a) and (b) are required only for the purpose of the proof given below and is not practically required.

(b) Treatment of a directed circuit

It has been found from experience that if there is a directed circuit, then the same term could lead to a resultant input back into it having a contradictory truth value, so that the argument is self-contradictory. This is obviously not desirable. Therefore, whenever a directed circuit occurs, it has to be broken at a suitable location, and the outgoing and incoming truth values of the same term have to be separated, and a vidya check incorporated between them. In our formalism, a logical graph will be checked for directed circuits and the question will be raised as to where the circuit is to be broken. This has to be stated in the beginning of the problem. Then, the reconstruction of the broken region is indicated in Fig.4(a). The directed circuit is indicated by full lines and the term $\underline{t}_1$ is both the initial input, and the final output from Z_N. Since these can be contradictory, we replace $\underline{t}_1$ by $\underline{t}'_1$ and $\underline{t}''_1$

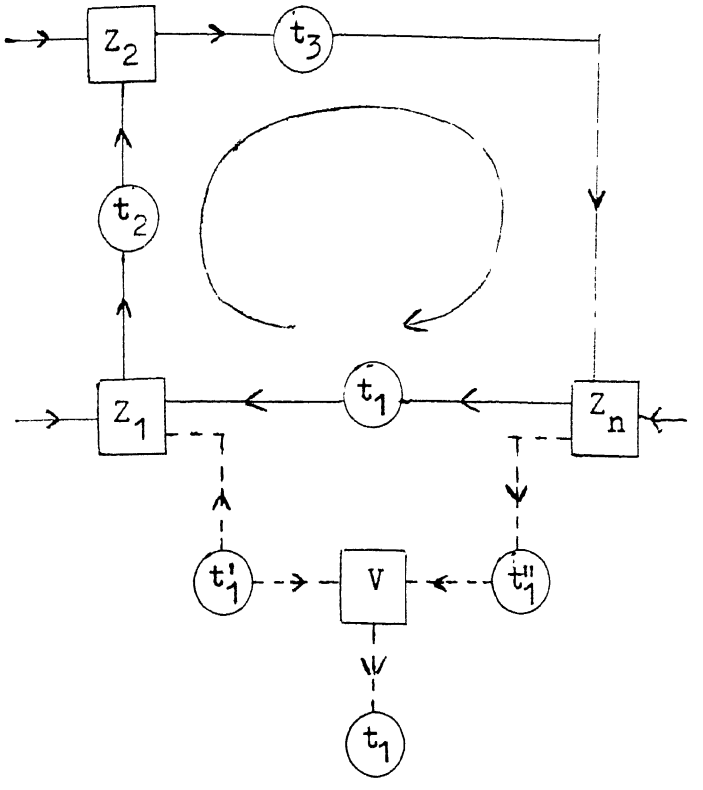


Fig.4(a). Reconstruction (in dotted lines) for removing a single directed circuit. Note that an "input" $\underline{t}'_{=1}$ and an "output" $\underline{t}_{=1}$ are produced thereby.

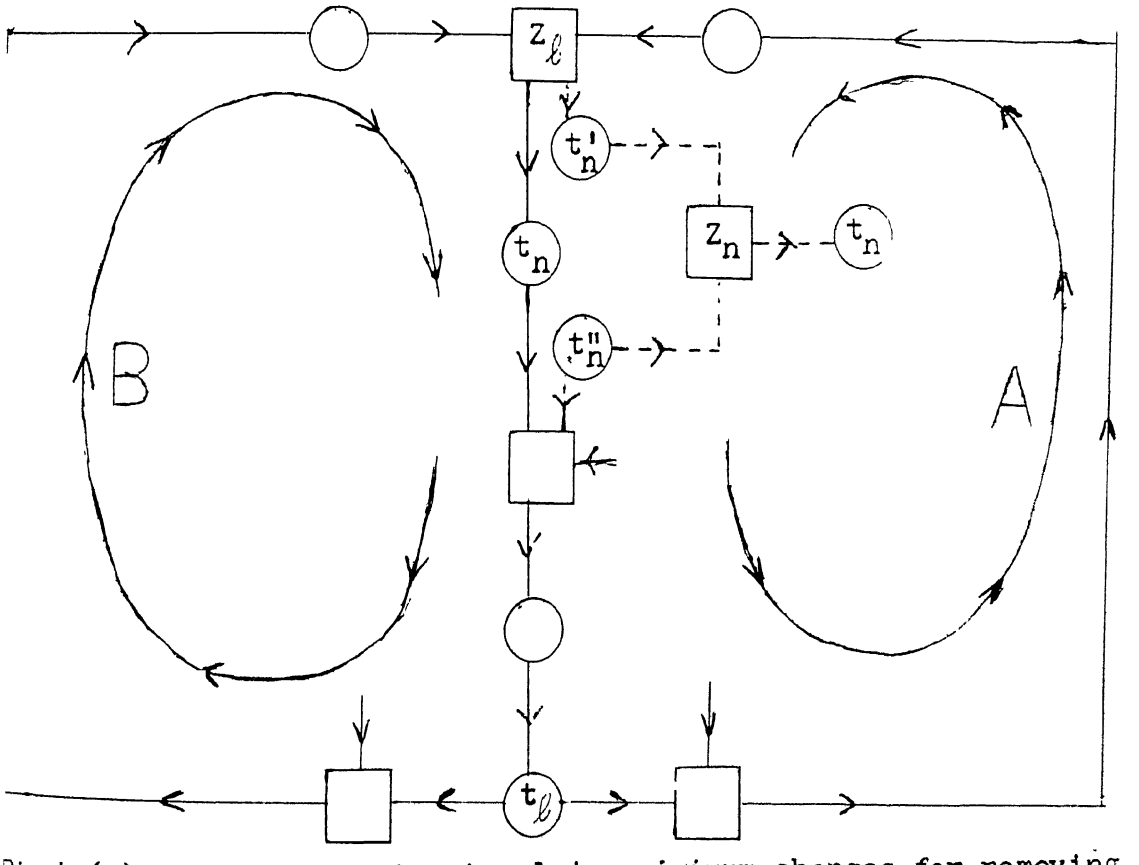


Fig.4.(b): Reconstruction involving minimum changes for removing two linked directed circuits A and B. One input $\underline{t}''_{=n}$ and one output $\underline{t}_{=n}$ are produced by this process.

where $\underline{t}'_1$ is the input and $\underline{t}''_1$ is the output from Z_N and we introduce the vidya check between $\underline{t}'_1$ and $\underline{t}''_1$ leading to the output $\underline{t}_1$. This check will give the information whether there is a contradiction or not. As will be seen clearly, the resultant circuit can still be built up of the two building blocks mentioned earlier.

It will be noticed that $\underline{t}'_1$ in Fig. 4(a) has no edges leading into it, so that it is an "input term". Similarly $\underline{t}_1$ has no edges leading out of it, so that it is an "output term." This property, that there is at least one input term and one output term in its graph, is a general property of all logical graphs, provided they do not have any directed circuits. We have seen a clear demonstration of this in Fig. 4(a) containing a single directed circuit. If there are two directed circuits, then it has only one structure as shown in Fig. 4(b), (provided only two edges can at most lead into a connective and only one edge leads out of it, while only one edge leads into a term and at most two edges lead out of it). In Fig. 4(b), these conditions are required at the connective $\underline{Z}_\ell$, at the beginning of the common link between the two circuits, and the term $\underline{t}_\ell$, at the end of this common link. The other terms and connectives

can have either two edges or three edges leading into, or away, from them, as is permissible. The two circuits A and B that are marked have a common path in the link from $\underline{\underline{Z}}_{\ell}$ to $\underline{\underline{t}}_{\ell}$.

In this case, the minimum change, needed to remove the directed circuits A and B, is indicated in Fig. 4(b), and it will be seen that there is one input term $(\underline{\underline{t}}''_n)$ and one output term $(\underline{\underline{t}}_n)$. Since the breaking of more than two circuits will lead only to a larger number of input and output terms, we shall take it as a general property that a logical graph containing N connectives has at least one input term and one output term. This is very useful for the proof of sequential implementability given in the next section.

(c) Proof of sequential implementability

With the above reconstructions made in the logical graph of an argument, we shall give a proof by induction that a graph $\mathcal{G}_N$ having N connectives, $\underline{\underline{Z}}_1$ to $\underline{\underline{Z}}_N$, each leading to a single output, $\underline{\underline{t}}_1$ to $\underline{\underline{t}}_N$, can be formulated in terms of a set of successively implementable elementary statements, each of the unary or binary form.

First we break up the given set of statements, in arbitrary sequence, into elementary statements containing only a unary or binary connective $\underline{\underline{Z}}$, and with only one edge from any connective upto a term. It can be verified that, for a small number of connectives, say 4 or 5, every graph that can be constructed using them (with the limitations mentioned above), can be represented by a sequentially implementable set of statements. We generalize this to $\mathcal{G}_N$ as follows, by showing that, if all graphs $\mathcal{G}_N$ are sequentially implementable then all graphs $\mathcal{G}_{N+1}$ are also so. The essential requirements for this proof is that every graph $\mathcal{G}_N$ has at least one input term and one output term (particularly the latter), a fact which was shown above for a general graph provided it does not have only directed circuits. If there are non-directed circuits and (or) only a tree structure, then there is always an output term in it. Thus, $\mathcal{G}_{N+1}$ will always have one output term at least, which we shall label as $\underline{\underline{t}}_{N+1}$, coming from the connective $\underline{\underline{Z}}_{N+1}$.

The connective $\underline{\underline{Z}}_{N+1}$ may be unary, as in Fig. 5(a), or binary, as in Fig. 5(b) or 5(c), and accordingly, it will have edges leading from one term $\underline{\underline{t}}_n$, or two terms $\underline{\underline{t}}_m$, $\underline{\underline{t}}_n$ as the

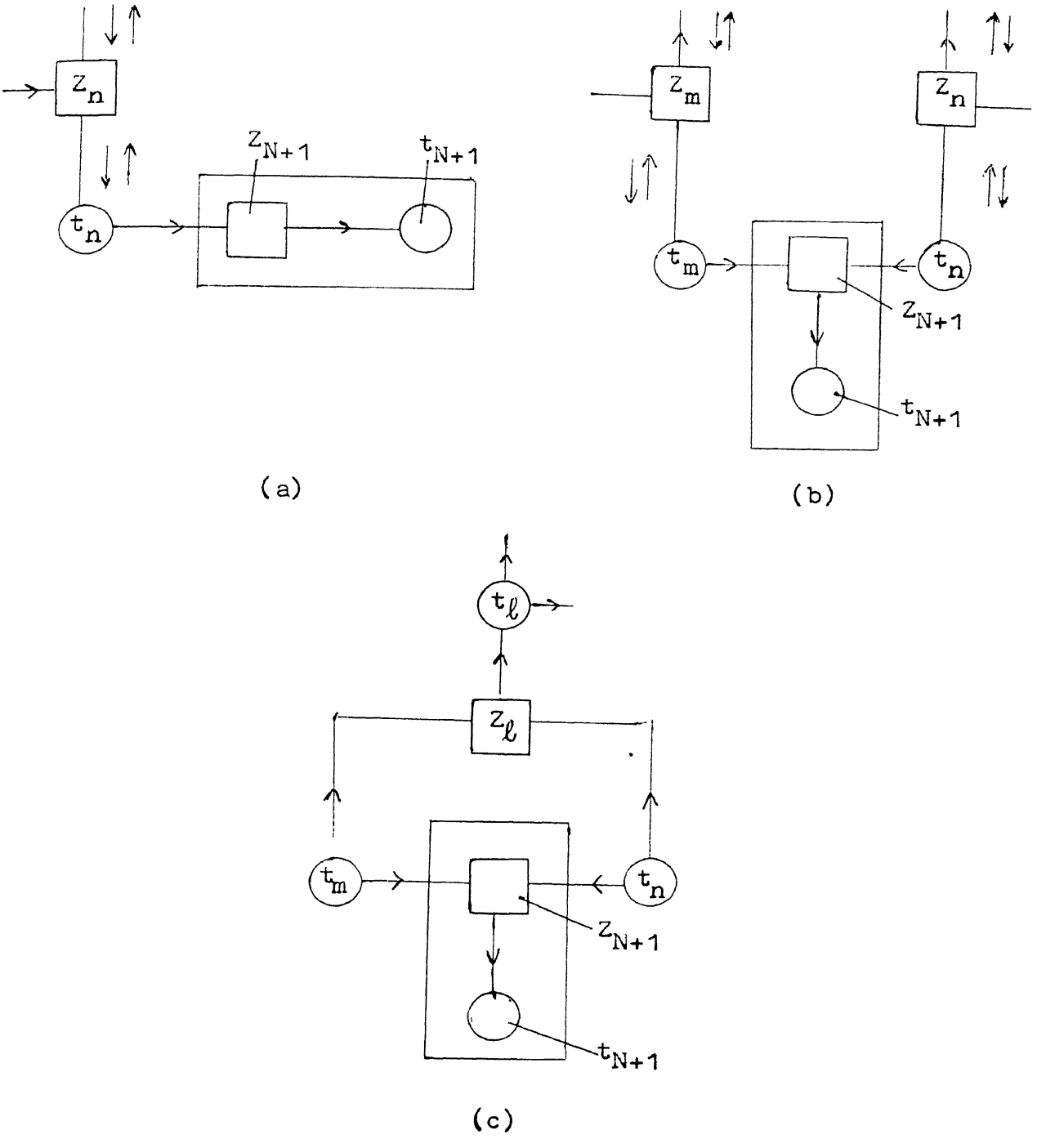


Fig.5. The block consisting of (Z_{N+1} and t_{N+1}) in $\mathcal{G}_{N+1}$, whose removal leaves $\mathcal{G}_N$ with at least one output term — t_N for unary Z_{N+1} in (a) and t_m, t_n for binary Z_{N+1} in (b). Both t_m and t_n are available in the listing of statements corresponding to graph $\mathcal{G}_N$ and the equation for the added block can be appended to it for $\mathcal{G}_{N+1}$.

case may be. In all cases, the different possible ways in which the rest of the graph $\mathcal{G}_N$ may be connected to $\underline{Z}_{N+1}$ are indicated in Fig.5. It will be seen that when $\underline{Z}_{N+1}$ and $\underline{t}_{N+1}$ are removed, the remaining graph is connected, and has all the properties mentioned above for a logical graph. Also, the equation connecting the inputs and output of $\underline{Z}_{N+1}$ can be written at the end of the list of sequentially implementable statements for $\mathcal{G}_N$, and can be implemented, because either $\underline{t}_n$, or $\underline{t}_m$, $\underline{t}_n$, are available as outputs in the list for $\mathcal{G}_N$.

Hence, if the graph $\mathcal{G}_N$ is sequentially implementable, then the additional step involving the connective Z_{N+1} in the graph $\mathcal{G}_{N+1}$ always so that $\mathcal{G}_{N+1}$ is sequentially implementable. is implementable. This is true for every $\mathcal{G}_{N+1}$, provided every $\mathcal{G}_N$ is sequentially implementable. Since this property is true for small values of N, it follows by induction that it is true for all values of N.

This proof is remarkably simple, but it seems to be very strong and, in fact, it can be converted into an algorithmic procedure for forward implementation, as will be described below. It is to be noted that although the proof is given starting from N+1 backwards, the existence of implementability in the forward direction is what is proved.

3. Implementation of a logical graph in general

(a) Algorithm for the forward implementation

This can be divided into two parts (i) Procedure prior to the conversion of an arbitrarily listed sequence into a sequentially implementable set of statements (SIS) and (ii) implementation of the sequentially implementable list to obtain the outputs of the graph for given inputs. The third part, namely reversal of the argument from the contradiction, if any back to its source is reserved for the next section.

The steps involved in this procedures are as follows.

1. The logical statements in the argument, as given, are first converted into elementary statements, and they are then listed in arbitrary order. The graph is constructed by making all the terms and connectives as the nodes of the graph, with directed edges interconnecting these.

2. The terms are then analysed and divided into three categories, namely input, intermediate term, and output, which are defined as follows. Input terms have no edges leading to them, and output terms have no edges leading out of them, while intermediate terms have edges both leading to them and leading away from them.

3. The graph is checked for directed circuits and, if any are present, the exact location at which each circuit is to be broken is decided, and a vidya check introduced between the two terms produced at each point of breakage.

4. Similarly, vidya checks are introduced wherever there are two or more inputs into a term as in Section 2(a)(ii).

5. So also, vidya checks are introduced at every binary reverse relation as in Section 2(a)(i).

6. At the end of this process, we will obtain a graph containing only non-directed circuits and consisting of only unary and binary relations.

Rearrange the modified set of equations for the graph so obtained so that they are sequentially implementable (the algorithm for this will be generalized after giving some examples below). When this is done, the argument will be implemented in a series of stages, at each of which a finite number of intermediate terms or final outputs are obtained, requiring only inputs from the initial input terms, or intermediate terms coming out in the stages prior to it. If, at any stage,

one of the intermediate terms comes out to be in the contradictory state X, then the implementation is stopped at the end of that stage, and the graph is checked in the reverse direction. The algorithm for this again will be given after working out examples for this purpose. As will be seen from these examples, this process will lead to the detection of those inputs which were responsible for the contradiction, in the form of conditions necessary for avoiding the contrandiction being checked

7. If, at any particular stage, all the intermediate terms are non-contradictory, then they are fed, along with the outputs of previous stages, into the equations for the next stage. This is continued until the listing of the final outputs is completed.

This method of dividing the initial list of elementary statements, in arbitrary order, into a sequential set of stages is very useful for parallel processing, and, in fact, plays an important part in the back-tracking process to be carried out in a systematic and unambiguous way. We shall first indicate the application of these ideas in problem solving, and extend these for theorem proving in a later section.

(b) Simple example of a logical graph illustrating the processes in the previous section

Table 1 lists a set of seven statements in arbitrary order forming the graph shown in Fig. 6. In this, we have an example of binary reverse relation, namely $\underline{\underline{h}}\ \underline{\underline{A}}\ \underline{\underline{k}} = T$, which has been broken up into the three equations (Sl. Nos. 4, 5 and 6), namely $\underline{\underline{h}}\ \underline{\underline{A}}\ \underline{\underline{k}} = \underline{\underline{x}}'$, $T = \underline{\underline{x}}''$, $\underline{\underline{x}}'\ \underline{\underline{V}}\ \underline{\underline{x}}'' = \underline{\underline{x}}$. It also has one non-directed circuit, consisting of two directed paths, both starting from the input $\underline{\underline{b}}$ and ending in a connective $\underline{\underline{A}}$. In the classification of this list into successive stages, we take, at each stage, those input term(s) of a relation which are available in the list of initial inputs and the intermediate terms output at the previous stages. If both (or only one) for a binary (unary) relation are available in this sublist, they are put together, as in 1.($\underline{\underline{a}}$,$\underline{\underline{b}}$) in stage 1, and the output of the relation $\underline{\underline{g}}$ marked therein. In the particular example, stage 1 can implement Sl. Nos 1, 3 and 5 of the arbitrary list. Then, these are carried over to stage 2, when Sl. Nos 2 and 7 become implementable. However, Sl. Nos 4 and 6 have only one of the two inputs needed in the available terms for the previous stages. These are then marked as ($\underline{\underline{h}}$, —) and ($\underline{\underline{x}}''$, —) and marked "Hold".

Table 1. Reorganizing an arbitrarily ordered set of statements of the logical graph of an argument into a set of sequentially implementable statements (SIS).

(a) Classification in stages

Sl. No.	Arbitrary listing *	Stage 1	Stage 2	Stage 3	Stage 4
1	$\underline{a}\ \underline{A}\ \underline{b} = \underline{g}$	1. $(\underline{a}, \underline{b}), \underline{g}$			
2	$\underline{g}\ \underline{I} = \underline{k}$	—	2. $(\underline{g}), \underline{k}$		
3	$\underline{c}\ \underline{O}\ \underline{b} = \underline{h}$	3. $(\underline{c}, \underline{b}), \underline{h}$			
4	$\underline{h}\ \underline{A}\ \underline{k} = \underline{x}'$	—	4. $(\underline{h}, —)$, Hold	4. $(\underline{h}, \underline{k}), \underline{x}'$	
5	$T = x''$	5. $(T), \underline{x}''$			
6	$\underline{x}'\ \underline{V}\ \underline{x}'' = \underline{x}$	—	6. $(\underline{x}'', —)$, Hold	6. $(\underline{x}'', —)$, Hold	6. $(\underline{x}'', \underline{x}'), \underline{x}$
7	$\underline{h}\ \underline{I}\ \underline{N} = \underline{y}$	—	7. $(\underline{h}), \underline{y}$		

* Inputs: $\underline{a}$, $\underline{b}$, $\underline{c}$, $\underline{x}'$; outputs $\underline{x}$, $\underline{y}$; Intermediate terms: $\underline{g}$, $\underline{h}$, $\underline{k}$, $\underline{x}''$

(b) Rearranged SIS

$\underline{a}\ \underline{A}\ \underline{b} = \underline{g}$ (1.1), $\underline{c}\ \underline{O}\ \underline{b} = \underline{h}$ (1.2), $T = \underline{x}''$ (1.3)

$\underline{g}\ \underline{I} = \underline{k}$ (2.1), $\underline{h}\ \underline{I}\ \underline{N} = \underline{y}$ (2.2)

$\underline{h}\ \underline{A}\ \underline{k} = \underline{x}'$ (3.1)

$\underline{x}'\ \underline{V}\ \underline{x}'' = \underline{x}$ (4.1)

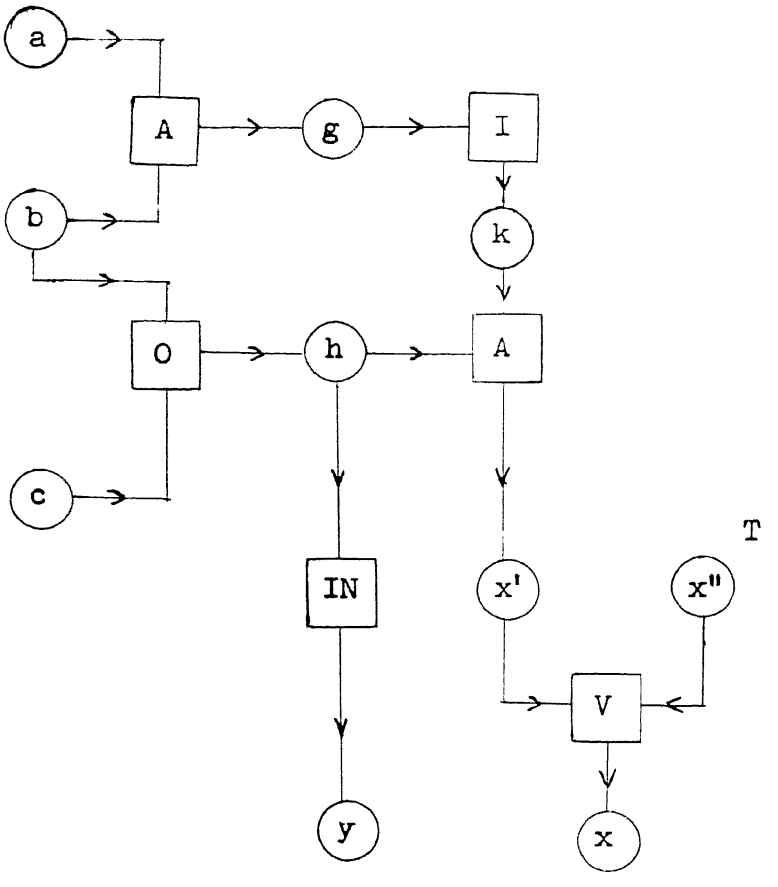


Fig.6 : An example of the logical graph of an argument in PC

Thereafter, only equations which are unclassified, and those which are under "Hold", need be considered for successive stages. When this is done, Eq.(4) becomes implementable in stage 3, while Eq.(6) continues to be "Hold", and is implemented only in stage 4.

Having made this classification and rearrangement, the sequentially implementable set is conveniently listed as in Table 1(b). Then, in their practical implementation, all the inputs of the statements in one row can be worked out in parallel, while the successive rows have to be implemented sequentially one after the other. The procedure indicated here is obviously quite general, and is applicable so long as there are no directed circuits in the graph. Essentially, we have extended a similar theorem, which is well-known in standard logic for trees, to the case when there are also non-directed circuits.

Using the sequentially implementable set in Table 1(b), the argument in the graph of Fig.6 has been worked out for the complete set of truth values for the inputs $\underline{\underline{a}}$, $\underline{\underline{b}}$, $\underline{\underline{c}}$, $\underline{\underline{x}}''$. Since $\underline{\underline{x}}''$ is given to be T for the binary reverse relation, it is not changed, while eight possible combinations of T, F for $\underline{\underline{a}}$, $\underline{\underline{b}}$, $\underline{\underline{c}}$, are listed.

It will be noticed that two of these inputs, Sl.Nos 4 and 8, lead to the contradictory state X for $\underline{\underline{x}}$, showing that they are not permissible for this argument. Suppose, one of these had been taken first for trial and the output $\underline{\underline{x}} = X$ / had been obtained in stage 4, then it is possible to reverse the argument from this and obtain the consequences by the procedures given in Table 2(b). We shall indicate this very briefly and reserve a more detailed discussion for a more complicated example in the succeeding sections.

(c) Reversal from a contradiction in the simple example

There is only one Eq.(4.1) in stage 4 giving the output and this is $\underline{\underline{x}} = X$, produced by the vidya check between $\underline{\underline{x}}'$ and $\underline{\underline{x}}''$. In both the examples, $\underline{\underline{x}}' = F$, $\underline{\underline{x}}'' = T$, leading to the contradiction. Since $\underline{\underline{x}}''$ cannot be changed, the only consequence of the reversal of (4.1) is that $\underline{\underline{x}}'^{(r)} = T$ as shown in the top row of Table 2(b). $\underline{\underline{x}}'^{(r)}$ is the output of the only relation in stage namely (3.1). Reversing this, we obtain the data in the first row against 3.1^r, namely that "$\underline{\underline{h}}^{(r)} \wedge \underline{\underline{k}}^{(r)} = T$", which necessarily requires that $\underline{\underline{h}}^{(r)} = T$, and $\underline{\underline{k}}^{(r)} = T$. We consider each of these in turn. As shown in (a), the former one, $\underline{\underline{h}}^{(r)} = T$

Table 2. Implementation for all possible inputs T, F for the logical graph in Fig. 6. (The SIS for this is given in Table 1(b)).

(a) Complete list of intermediate terms and outputs in the forward direction of implementation

Sl No	Inputs $\underline{a}$	$\underline{b}$	$\underline{c}$	$\underline{x}''$	Stage 1 $\underline{g}$	$\underline{h}$	Stage 2 $\underline{k}$	$\underline{y}$	Stage 3 $\underline{x}'$	Stage 4 $\underline{x}$
1	T	T	T	T	T	T	T	F	T	T
2	T	T	F	T	T	T	T	F	T	T
3	T	F	T	T	F	T	D	F	D	T
4	T	F	F	T	F	F	D	D	F	X
5	F	T	T	T	F	T	D	F	D	T
6	F	T	F	T	F	T	D	F	D	T
7	F	F	T	T	F	T	D	F	D	T
8	F	F	F	T	F	F	D	D	F	X

(b) Reversing from $\underline{x} = X$ in Stage 4

4.1^r: $\underline{x} \neq X \longmapsto x'^{(r)} = T$

3.1^r: $\underline{x}'^{(r)} = T \longmapsto (\underline{h}^{(r)} = T) \wedge (\underline{k}^{(r)} = T)$ (a), (b)

(a) $\underline{h}^{(r)} = T \longmapsto \underline{h} \vee \underline{h}^{(r)} = X$, so that

1.2^r: $\underline{h}^{(r)} = T \longmapsto (\underline{c}^{(r)} = T) \vee (\underline{b}^{(r)} = T)$

(b) $\underline{k}^{(r)} = T \longmapsto \underline{k} \vee \underline{k}^{(r)} = T \neq X$, so that no changes are demanded for $\underline{g}$ in (2.1), and hence for $\underline{a}$, $\underline{b}$ in (1.1).

Hence, if $\underline{x} \neq X$, then $\underline{b}^{(r)} = T$ or $\underline{c}^{(r)} = T$

Table 2(c): Reversing from the purification of $\underline{x}' = D$ from $\underline{x} = T$*

Sl No	Inputs a	b	c	x"	Stage 1 g	h	Stage 2 k	y	Stage 3 x'	Stage 4 x
3	T	F	T	T	F	T	D	F	D	T
	"	"	"	"	[F]	T	(T)	F	(T)	(T)

$$\underline{x}' \ \underline{V} \ \underline{x}'' = \underline{x}^r = T \longmapsto \underline{x}'^r = T \qquad (4.1)^r$$

$$\underline{h} \ \underline{A} \ \underline{k} = \underline{x}'^r = T \longmapsto \underline{k}^r = T \qquad (3.1)^r$$

$$\underline{g} \ \underline{I} = \underline{k}^r = T \longmapsto \underline{k}^r \ \underline{N} \ \underline{I} \ \underline{N} = \underline{g}^r = F \qquad (2.1)^r$$

$\underline{g}^r \ \underline{V} \ \underline{g} = F$ (unchanged) "absorbed"

* Text in Section 4(d)

Table 2. Implementation for all possible inputs T, F for the logical graph in Fig. 6. (The SIS for this is given in Table 1(b)).

(a) Complete list of intermediate terms and outputs in the forward direction of implementation

Sl No	Inputs $\underline{a}$	$\underline{b}$	$\underline{c}$	$\underline{x}''$	Stage 1 $\underline{g}$	$\underline{h}$	Stage 2 $\underline{k}$	$\underline{y}$	Stage 3 $\underline{x}'$	Stage 4 $\underline{x}$
1	T	T	T	T	T	T	T	F	T	T
2	T	T	F	T	T	T	T	F	T	T
3	T	F	T	T	F	T	D	F	D	T
4	T	F	F	T	F	F	D	D	F	X
5	F	T	T	T	F	T	D	F	D	T
6	F	T	F	T	F	T	D	F	D	T
7	F	F	T	T	F	T	D	F	D	T
8	F	F	F	T	F	F	D	D	F	X

(b) Reversing from $\underline{x} = X$ in Stage 4

4.1^r: $\underline{x} \neq X \longmapsto x'^{(r)} = T$

3.1^r: $\underline{x}'^{(r)} = T \longmapsto (\underline{h}^{(r)} = T) \wedge (\underline{k}^{(r)} = T)$ (a), (b)

(a) $\underline{h}^{(r)} = T \longmapsto \underline{h} \vee \underline{h}^{(r)} = X$, so that

1.2^r: $\underline{h}^{(r)} = T \longmapsto (\underline{c}^{(r)} = T) \vee (\underline{b}^{(r)} = T)$

(b) $\underline{k}^{(r)} = T \longmapsto \underline{k} \vee \underline{k}^{(r)} = T \neq X$, so that no changes are demanded for $\underline{g}$ in (2.1), and hence for $\underline{a}$, $\underline{b}$ in (1.1).

Hence, if $\underline{x} \neq X$, then $\underline{b}^{(r)} = T$ or $\underline{c}^{(r)} = T$

4. Additional features of implementation of logical graphs

(a) Combination of a string of unary relations into a simple one

There are three features dealing with this aspect which must be pointed out. First, in the preparation of the graph prior to its implementation, but after the graph has been constructed and analysed, if there is a string of successive unary relations, they must be combined and reduced to a single unary relation. This is indicated in Fig. 7.

The next two procedures are in relation to the implementation of the graph.

(b) Reversal from a "purification" produced by the vidya check

In the last section, we have briefly described the procedure for reversing the direction of the progress of the implementation of an argument in a logical graph, from a contradiction,back to the term which is the source of the contradiction, which can then be given the opposite truth value to avoid the contradiction. In fact, it is necessary to do a similar reverse back-tracking process when the doubtful state gets "purified" to a pure state, T or F. We shall use the symbol P (standing for "pure"), to indicate either of the states T or F. As is quite clear, a binary forward relation with inputs which are pure can only lead to a pure state as output, so that the doubtful state can arise

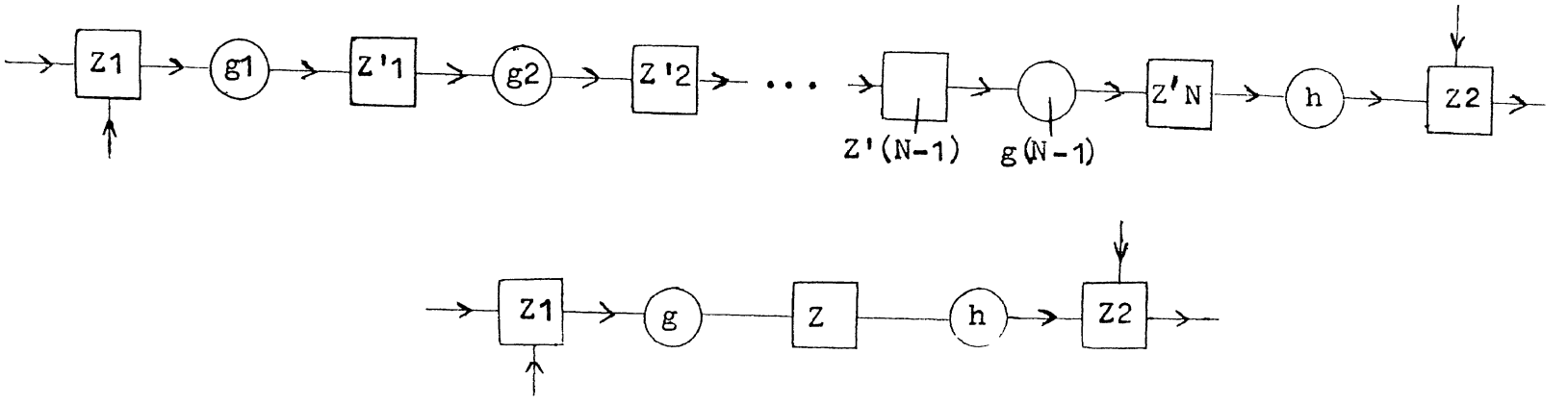


Fig.7. Reduction of a string of unary relations to a single relation. The relevant matrix is $|Z| = |Z'1|Z'2| \ldots |Z'N|$, where $|Z'j|$ is one of $|\underline{\underline{I}}(k, \ell)|, |\underline{\underline{E}}(k, \ell)|$ or $|\underline{\underline{D}}|$.

only as the output of a unary relation, if the initial inputs to the graph are all pure states as we have indicated earlier. Therefore, this doubtful state, which has been produced as the output of a unary relation, will propagate forward through binary and unary relations, but may get "absorbed" at a binary relation whose output may be pure, although one of the inputs may be doubtful. Examples are, $F \underline{\underline{A}} D = F$, $T \underline{\underline{O}} D = T$. On the other hand, it may also go through a binary relation as a doubtful state as in $T \underline{\underline{A}} D = D$, $F \underline{\underline{O}} D = D$. In such a case, if the output $\underline{\underline{c}}'$ of a binary relation $\underline{\underline{a}} \underline{\underline{Z}} \underline{\underline{b}} = \underline{\underline{c}}'$, is to be checked for contradiction with another input $\underline{\underline{c}}''$ to the same term, via the vidya check $\underline{\underline{c}}' \underline{\underline{V}} \underline{\underline{c}}'' = \underline{\underline{c}}$, it is quite possible that $\underline{\underline{c}}''$ is pure so that $\underline{\underline{c}}' = D$ gets purified to $\underline{\underline{c}} = T$ or F . In such a case, it is obvious that the resultant pure state output has come out of, either of the two components T or F, of the doubtful state D, so that it is essential that this change from D to P be introduced at the earliest point at which the D-state arose. This can be done, by back-tracking from the particular vidya check for $\underline{\underline{c}}$ at which the purification took place, to the point of origin of the D-state, as will be explained in two examples below. In that case, the graph must be

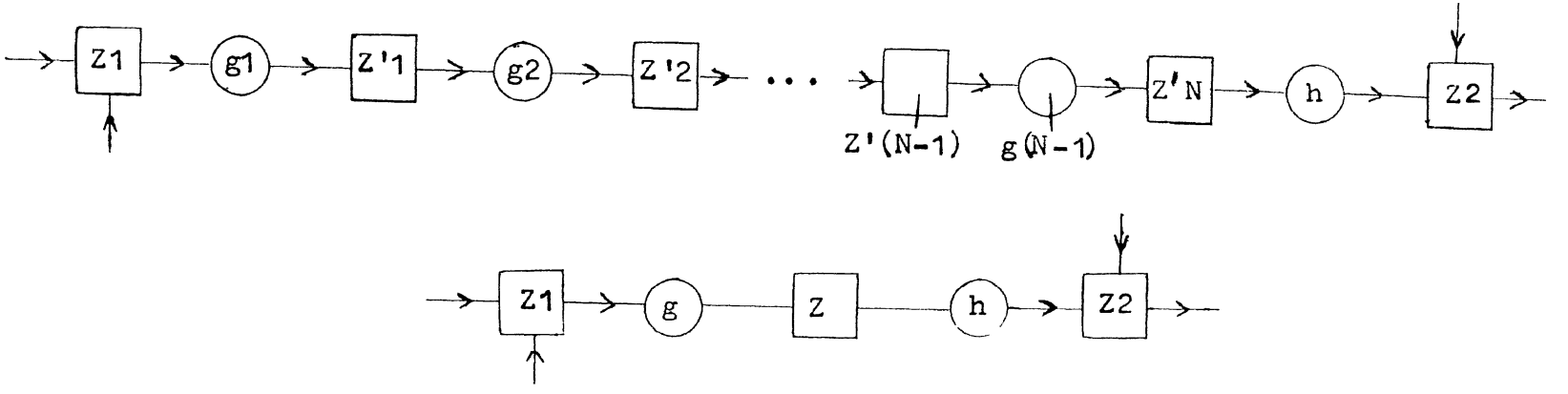


Fig.7. Reduction of a string of unary relations to a single relation. The relevant matrix is $|Z| = |Z'1|Z'2| \ldots |Z'N|$, where $|Z'j|$ is one of $|\underline{\underline{I}}(k, \ell)|, |\underline{\underline{E}}(k, \ell)|$ or $|\underline{\underline{D}}|$.

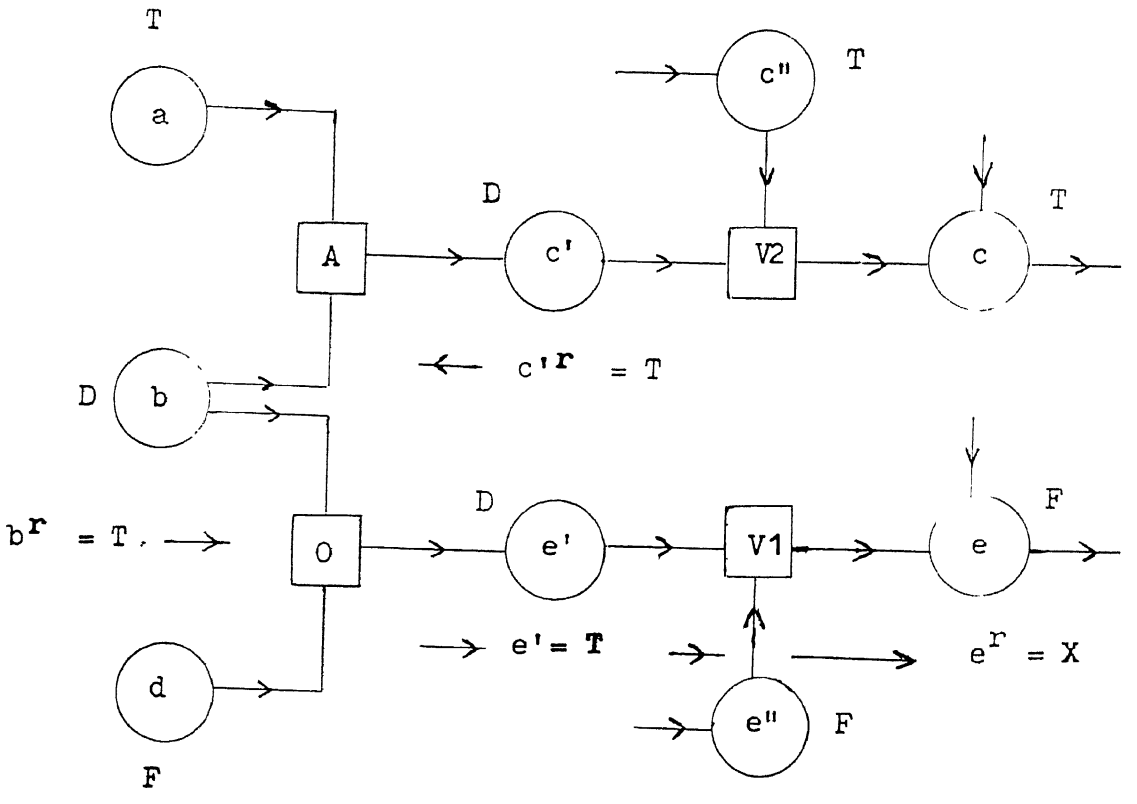


Fig.8(a): The doubtful state $\underline{c}' = D$ is converted to the pure state $\underline{c} = T$ by the vidya check **V2** . On reversing this, $\underline{c}'^{r} = T$, which leads to $\underline{b}^{r} = T$, and on back-tracking forward from $\underline{b}^{r}$, we obtain $\underline{e}'^{r} = T$, so that $\underline{e}^{r} = X$ (see text for further details).

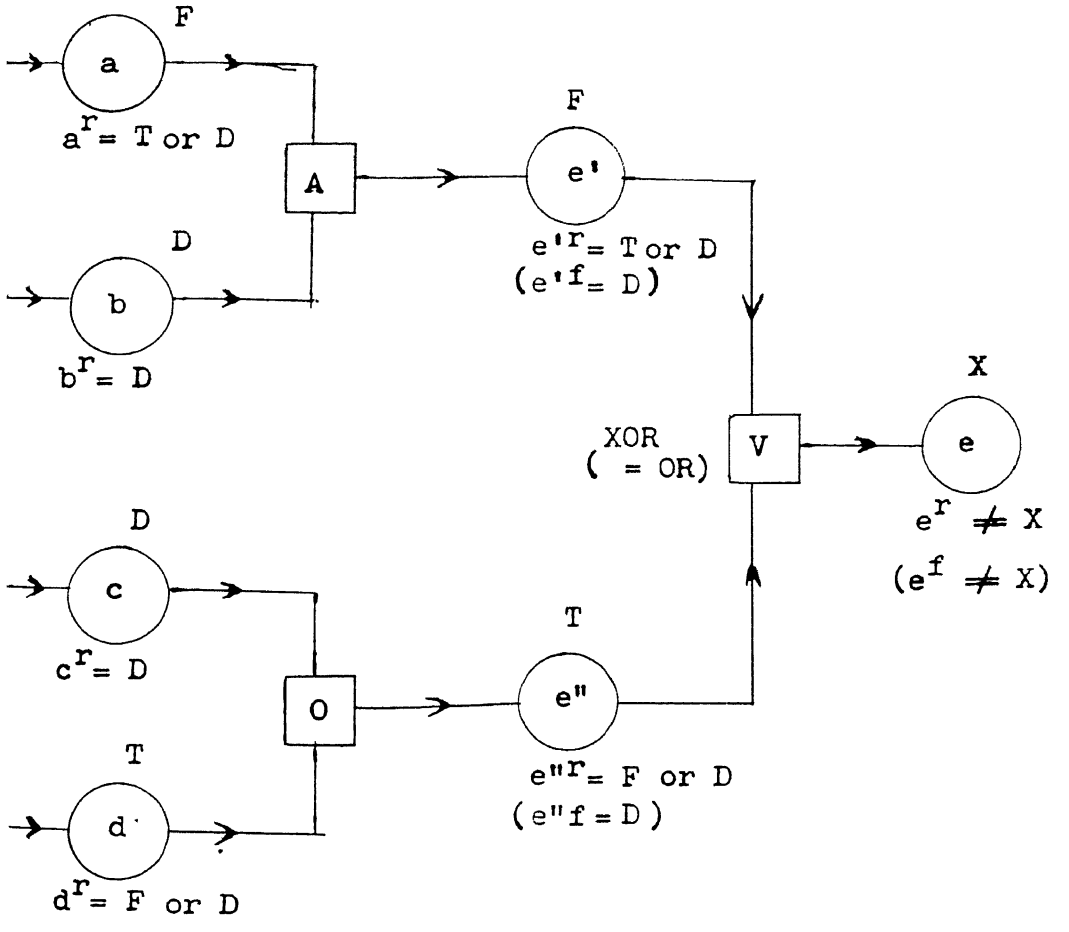


Fig.8(b): Special case of reversal from a contradiction X arising from a vidya check, when XOR is modified to OR . It is similar in pattern to Fig. 8(a). On reconstructing the argument in the forward direction from the reversed values we obtain $\underline{a}^r$ = T , D and $\underline{b}^r$= D lead to $\underline{e}'^f$= D, and $\underline{c}^r$= D, and $\underline{d}^r$= F,D lead to $\underline{e}"^f$= D, so that $\underline{e}^f \neq$ X, in either case or, when both changes have been made. (See text Section 4(d).)

mutually not consistent if the inputs $\underline{a}$, $\underline{b}$, $\underline{d}$ have the truth values as given. It is to be noted that this not a contradiction in the argument, but is so only for the particular inputs, and is a feature that must be taken care of in the implementation process of the logical argument.

(c) Simple example of reversal from purification of D (Graph of Fig 6)

This is indicated in Table 2(c). It will be noticed that D occurs for the first time for $\underline{k}$ in stage 2 of Table 2(a), Sl. Nos. 3, 5, 6, 7 (all of which are non-contradictory), and that it continues to be D in stage 3 also, while it gets purified to the state T by the vidya check in stage 4 (in all the cases). Therefore, the reversal from the purification from D to T in stage 4 can be carried out as shown in Table 2(c), for the typical example of Sl. No. 3. The third row from Table 2(a) is repeated at the top of this table, and the effect of the purification going backward from stage 4 to stages 3, 2 and 1 respectively are indicated by circles in the entries of the second row. The three equations that are reversed are as given in the second half of the table, namely (4.1), (3.1) and (2.1).

In $(4.1)^r$, the value of T for $\underline{\underline{x}}$ imposes the value of T for $\underline{\underline{x}}'$ This value, input in the reverse sense in $(3.1)^r$, leads to $\underline{\underline{k}}^r = T$ in Eq.$(3.1)^r$. This again, input in the reverse direction in $(2.1)^r$, leads to $\underline{\underline{g}}^r = D$. However, this doubtful state is consistent with the value $\underline{\underline{g}} = F$ in the forward direction and is therefore absorbed in it. Therefore, the reversal from the purification stops at that stage, as indicated by square ring round the entry for $\underline{\underline{g}}$.

We shall give a more detailed example in the next few sections.

(d) Special considerations relating to the reversal from an X produced by the vidya check

Just as some special considerations arise when a conjunction and a disjunction are inter-related to one another and both get modified during reversal from a purification, similar special circumstances can arise from the reversal from a vidya check which indicates contradiction (X). A typical example is shown in Fig. 8(b) where $\underline{\underline{a}}\ \underline{\underline{A}}\ \underline{\underline{b}} = F = \underline{\underline{e}}'$ and $\underline{\underline{c}}\ \underline{\underline{O}}\ \underline{\underline{d}} = T = \underline{\underline{e}}''$ and $\underline{\underline{e}}'\ \underline{\underline{V}}\ \underline{\underline{e}}'' = \underline{\underline{e}}$ has the contradictory state X. According to the standard procedure for reversing from a vidya check, either $\underline{\underline{e}}'^r$ must be changed from F, to T or D, or $\underline{\underline{e}}''$ has to be changed from T, to F or D. However, in the particular case where $\underline{\underline{b}}$, one of the inputs to $\underline{\underline{A}}$, and/or / $\underline{\underline{c}}$

one of the inputs to $\underline{\underline{Q}}$ are both D, and the other input in each case decides the output $\underline{\underline{e}}'$, or $\underline{\underline{e}}''$, as the case may be, on doing the reversal as mentioned above, we obtain $\underline{\underline{a}}^r$ = T or D, $\underline{\underline{b}}^r = \underline{\underline{c}}^r$ = D and $\underline{\underline{d}}^r$ = F or D. Now, if the same reversed truth values are input in the forward direction, we only obtain $\underline{\underline{e}}'^f$ = D and $\underline{\underline{e}}''^f$ = D. Therefore, the condition of "exclusive OR" is converted into "ordinary OR", and either of the reversals changin the input $\underline{\underline{e}}'$, or the input $\underline{\underline{e}}''$, put into the vidya check, is sufficient to make $\underline{\underline{e}} \neq X$. It can be readily verified that this is independent of $\underline{\underline{b}}$ and $\underline{\underline{c}}$ being doubtful states coming via separate paths, or coming from the same original source. In fact, a sufficient condition for XOR to be replaced by OR is that one of the four inputs $\underline{\underline{a}}$, $\underline{\underline{b}}$, $\underline{\underline{c}}$, $\underline{\underline{d}}$ in Fig. 8(b) is D.

The conditions $\underline{\underline{e}}'^r$ = T, D and $\underline{\underline{e}}''^r$ = F, D are equivalent to $\underline{\underline{e}}'^r \neq F$, $\underline{\underline{e}}''^r \neq T$. These are not states in BA-2, but require BA-3 for their description. A rigorous discussion of this is reserved for the next lecture.

5. A more complicated example of a logical graph

An example of an argument whose logical graph contains a number of interconnected non-directed circuits is shown in Fig.9. The elementary statements forming its steps are listed in arbitrary order in Table 3. It may be mentioned that in the full argument, there are two edges leading into $\underline{\underline{d2}}$ and three edges leading into $\underline{\underline{x}}$, both of which have been separated by suitable vidya checks. So also, there is a chain of two unary implications connecting $\underline{\underline{a3}}$ and $\underline{\underline{b2}}$, which have been replaced by a single one, namely $\underline{\underline{I}}(1, 2)$.

in stages

Table 3. Analysis of the listing of the logical argument in Fig.9

Inputs a1, a2, a3, a4

Sl No	Listing in arbitrary order	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6
1	a1 I = b1	(a1) b1					
2	a1 O a2 = c1	(a1, a2) c1					
3	a2 A b1 = c2	(a2) Hold	(a2, b1) c2				
4	a3 O(2,2) c1 = d1	(a3, —) Hold	(a3, c1) d1				
5	a3 I(1,2) = b2	(a3) b2					
6	d1 A b2 = d2"	—	(—, b2) Hold	(d1, b2) d2'			
7	a3 O a4 = d2'	(a3, a4) d2'					
8	d2' V d2" = d2	—	(d2', —) Hold	Hold	(d2', d2") d2		
9	a4 O d2 = e2	(a4, —) Hold	Hold	Hold	Hold	(a4, d2) e2	
10	d2" I = e1"	—	—	—	(d2") e1"		
11	c2 I = e1"	—	—	(c2) e1'			
12	e1' V e1" = e1,	—	—	—	(e1', —) Hold	(e1', e1") e1	
13	e1 V e2 = x	—	—	—	—	—	(e1, e2) e

Table 4 : SIS of the statement for the logical graph in Fig. 9

Inputs

a1 , a2 , a3 , a4

Stage 1

a1 I = b1 (1.1) ; a1 O a2 = c1 (1.2) ; a3 I(1,2) = b2 (1.3);
a3 O a4 = d2' (1.4)

Stage 2

a2 A b1 = c2 (2.1); a3 O(2, 2) c1 = d1 (2.2)

Stage 3

d1 A b2 = d2"(3.1); c2 I = e1' (3.2)

Stage 4

d2' V d2" = d2 (4.1) ; d2" I = e1" (4.2)

Stage 5

a4 O d2 = e2 (5.1) ; e1' V e1" = e1 (5.2)

Stage 6

e1 V e2 = x (6)

Table 5(a) Truth values obtained at various stages in the implementation of the logical graph in Fig. 9

Sl No	Inputs a1 a2 a3 a4	Stage 1 b1 c1 b2 d2'	Stage 2 c2 d1	Stage 3 d2" e1'	Stage 4 d2 e1"	Stage 5 e2 e1	Stage 6 x
1	T T T T	T T F T	T F	F T	(X)* D	- -	-
2	T T T F	T T F T	T F	F T	(X)* D	- -	-
3	T T (F) T	T T (D) T	T T	(D)† T	(T) D	T T	T
	F ←	T ←		T ←			
4	T T (F) F	T T (D) F	T T	(D)† T	(F) D	F T	X**
	D ←	F ←		F ←			
5	T F F F	T T D F	F T	D D	F D	F D	F
6	T F F T	T T D T	F T	D D	T D	T D	T
7	(F) (T) F F	(D) T (D) F	(D) T	(D) (D)	F (D)	F (D)	F
	D ←			F ←	(b) F ←	F ←	F
	F ←		F ←	(a) F ←			
8	F T F T	D T D T	D T	D D	F D	T D	T
9	F F F F	D F D F	F T	D D	F D	F D	F
10	F F F T	D F D T	F T	D D	T D	T D	T

* Reversal from X of Sl. Nos 1 and 2 (see Fig. 10(a))

** Reversal from X of Sl. No. 4 (see Fig. 10(b))

Table 5. Contd.

Table 5(b)
Reversal from D (indicated by rings) in Sl. Nos 3, 4 and 6

Sl. No. 3 : $\underline{d2'} \underline{V} \underline{d2''} = \underline{d2} = T \longmapsto \underline{d2''}^r = T$ $(4.1)^r$

$\underline{d1} \underline{A} \underline{b2} = \underline{d2'} \longmapsto \underline{b2}^r = T$ $(3.1)^r$

$\underline{a3} \underline{I}(1,2) = \underline{b2} \longmapsto \underline{a3}^r = \underline{b2}^r \underline{I}(1,2) = F$ $(1.2)^r$

(agrees)

Sl. No. 4 : $\underline{d2'} \underline{V} \underline{d2''} = \underline{d2} = F \longmapsto \underline{d2''}^r = F$ $(4.1)^r$

$\underline{d1} \underline{A} \underline{b2} = \underline{d2'} \longmapsto \underline{b2}^r = F$ $(3.1)^r$

$\underline{a3} \underline{I}(1,2) = \underline{b2} \longmapsto \underline{a3}^r = \underline{b2}^r \underline{I}(1,2) = D$ $(1.2)^r$

(consistent)

Sl. No. 6 : $\underline{e1} \underline{V} \underline{e2} = \underline{x} = F \longmapsto \underline{c1}^r = F$ $(6)^r$

$\underline{e1''} \underline{V} \underline{e1'} = \underline{e1} = F \longmapsto \underline{e1} = F$ (a), $\underline{e1'} = F$ (b) $(5.2)^r$

(a) $\underline{c2} \underline{I} = \underline{e1'} = F \longmapsto \underline{e1'}^r \underline{N} \underline{I} \underline{N} = \underline{c2}^r = F$ $(3.2)^r$

$\underline{a2} \underline{A} \underline{b1} = \underline{c2} = F \longmapsto \underline{b1}^r = F$ $(2.1)^r$

$\underline{a1} \underline{I} = \underline{b1} = F \longmapsto \underline{b1}^r \underline{N} \underline{I} \underline{N} = \underline{a1}^r = F$ $(1.1)^r$

(agrees)

(b) $\underline{d2''} \underline{I} = \underline{e1''} = F \longmapsto \underline{e1''}^r \underline{N} \underline{I} \underline{N} = \underline{d2''}^r = F$ $(4.2)^r$

$\underline{d1} \underline{A} \underline{b2} = \underline{d2''}^r \longmapsto \underline{b2}^r = F$ $(3.1)^r$

$\underline{a2} \underline{I}(1,2) = \underline{b2} = F \longmapsto \underline{b2}^r \underline{I}(1,2) = \underline{a2}^r = D$ $(1.3)^r$

(consistent)

(a) Reversal from contradictions to discover their source.

The procedure for listing this argument in a sequentially implementable sequence is shown in detail in Table 3 and the SIS of the statement is given in Table 4. The implementation is thereafter straightforward and can be carried out for any one of the 16 possible combinations of (T, F) for $\underline{\underline{a1}}$, $\underline{\underline{a2}}$, $\underline{\underline{a3}}$, $\underline{\underline{a4}}$. The truth value of all the intermediate terms, and output for the first four of these, are given in Table 5, where it will be seen that Sl. Nos. 1 and 2 lead to the contradictory state X for $\underline{\underline{d2}}$. The purpose of this section is to indicate in somewhat more detail the consequences of reversal, from the contradiction X, to obtain a set of conditions on the inputs which suffice for the contradiction to be removed. Taking, in particular, Sl. No. 1, the procedure of back-tracking from the contradiction in stage 4 is shown in Fig. 10* . We shall discuss this in some detail .

The contradiction has arisen from the vidya check between $\underline{\underline{d2}}'$ and $\underline{\underline{d2}}''$, which have opposite states T and F. Therefore, if the contradiction is to be avoided, either the first one has to be changed to F, or the second one changed to T. This is indicated by XOR in Fig.10 and the two possibilities are shown in the top row and in the third row. The former one, $\underline{\underline{d2}}'^{r}$ = F, goes back

---

* The considerations briefly put forward in Section 4(d) that reversal from a contradiction from a term $\underline{t}$ = T or F leads to $\underline{t}^{r}$ having the truth values $\neg\underline{t}$ or $\neg\underline{f}$ rather than F or T, has not been adopted in this section. It can be verified that this only leads to possible truth values D for $\underline{\underline{a1}}^{r}, \underline{\underline{a2}}^{r}, \underline{\underline{a3}}^{r}, \underline{\underline{a4}}^{r}$ in addition to those obtained here. Since inputs cannot have a D-state, they can be disregarded and the results given below are valid in 2-valued logic. However, the D-state cannot be omitted at intermediate stages, but only at the final input stage of the back-tracking process.

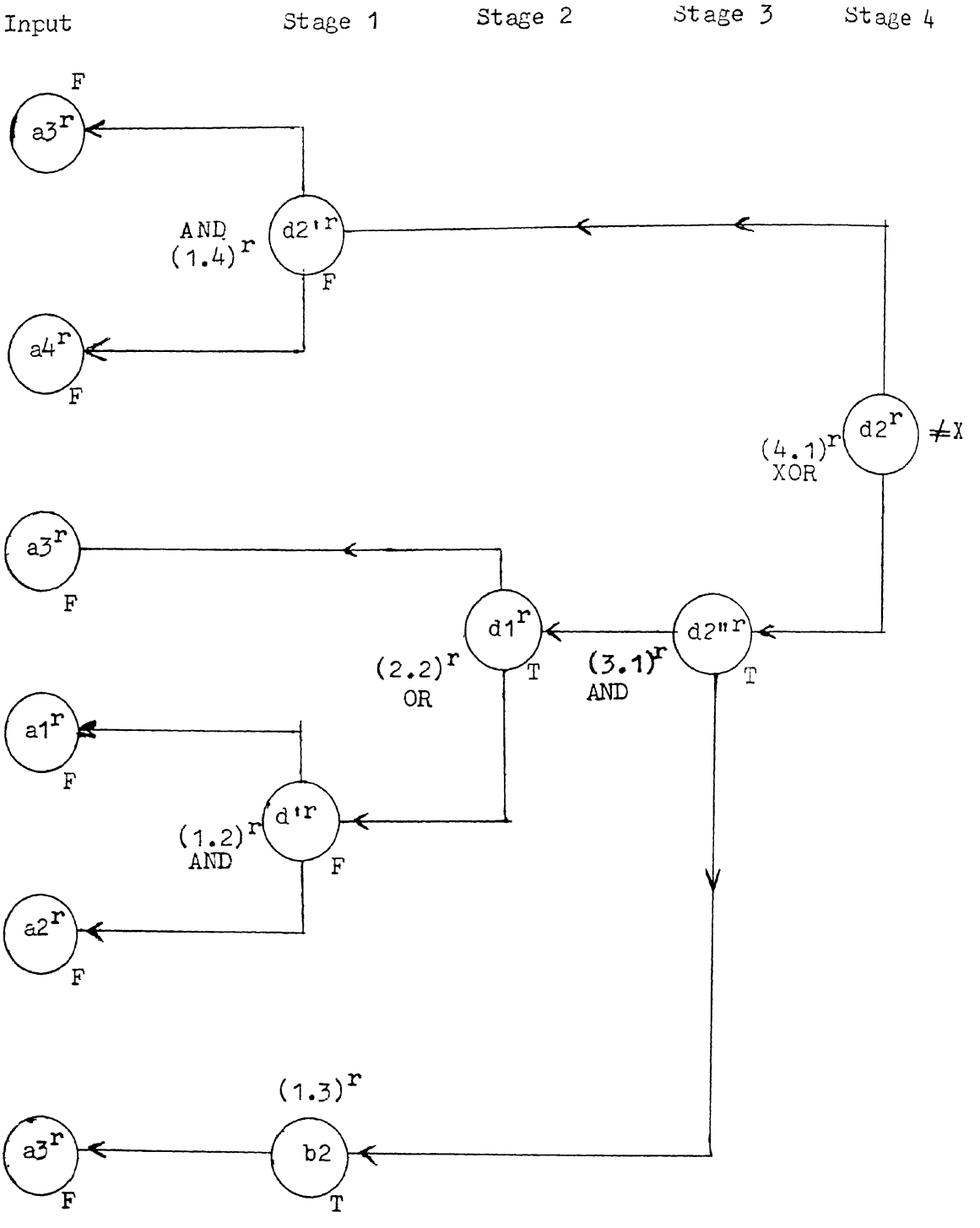


Fig.10(a): Summary of the back-tracking from a contradiction X in stage 4 for Sl. No. 1 of Table 5.

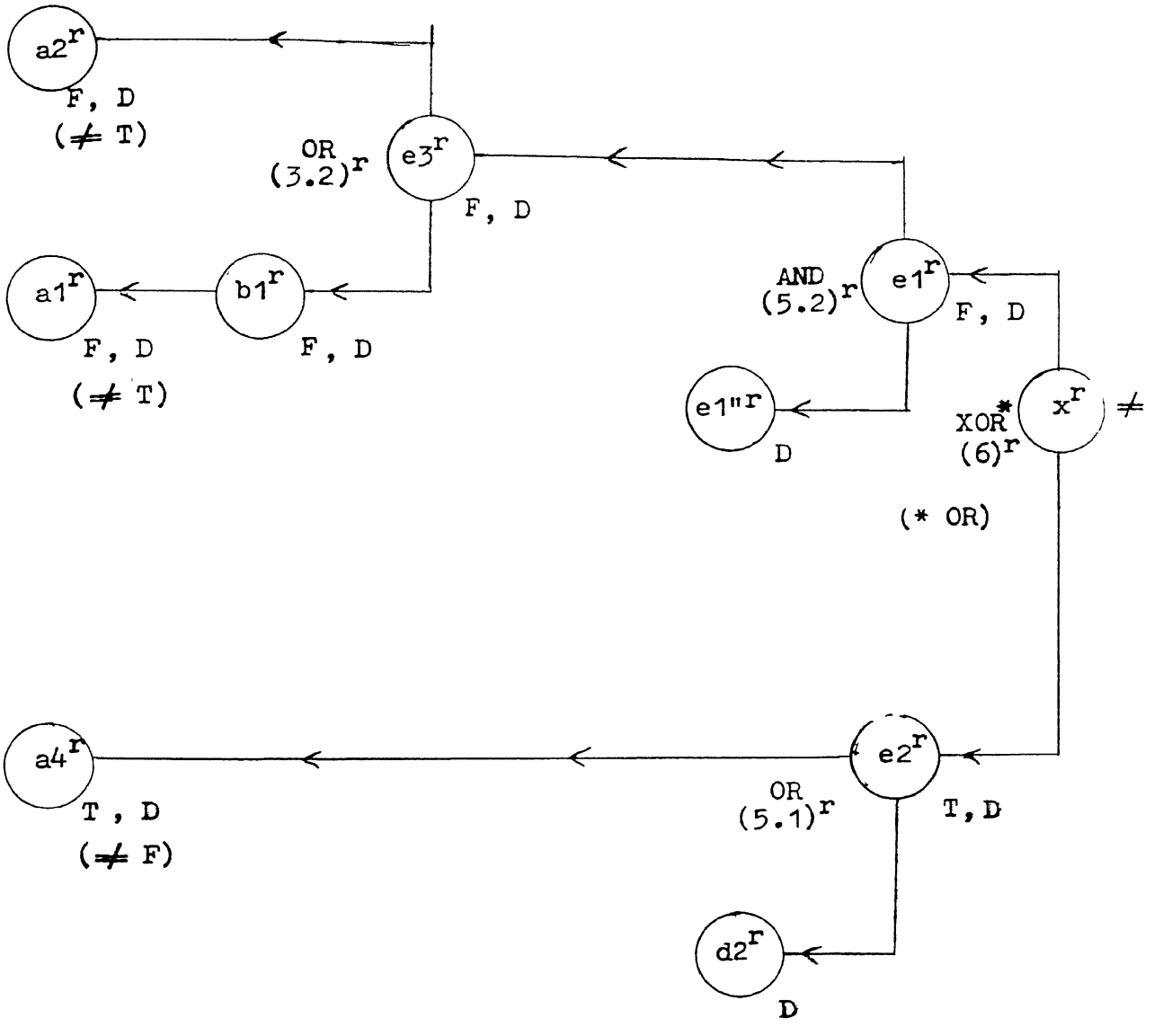


Fig.10(b): Summary of the back-tracking from a contradiction X in stage 6 for Sl. No. 4 . For the reasons given in Section 4(d), the XOR becomes OR . Also, the additional possibilities of D at the intermediate stages of the back-tracking is marked in this diagram alone.

right to stage 1, and on reversing (1.4), connecting $\underline{a3}$ and $\underline{a4}$ with $\underline{d2'}^{r}$, we obtain the conclusion $\underline{a3}^{r} = F$ and $\underline{a4}^{r} = F$. This is because a disjunction is negated and is equivalent to the conjunction of the individual negations.

The reversal from $\underline{d2'}^{r} = T$ leads to three branches, one at / an AND connective in $(3.1)^{r}$, another in an OR connective in $(2.2)^{r}$, and once again in an OR connective in $(1.2)^{r}$, which becomes / an AND because the disjunction is negated. Also, in the second part of the first branch, there is a unary relation $(1.3)^{r}$, which takes the back-tracking from stage 1 to the input stage. In each case, the effect on the conjunction, or disjunction, imposed by the reversal is taken to be given by the binary reverse relation, with the input being the truth value of the relation.

The sum total of the reversed values for the input can be given as follows:

$$(\underline{a3}^{r} = F \wedge \underline{a4}^{r} = F) \text{ XOR } ((\underline{a3}^{r} = F) \vee (\underline{a1}^{r} = F \wedge \underline{a2}^{r} = F)) \wedge (\underline{a3}^{r} = F) \tag{3.9}$$

Since $\underline{a3}^{r} = F$ is common to the right side and the left side of the XOR, this can be converted to a simple OR, so that (3.9) becomes

$$(\underline{a3}^{r} = F \wedge \underline{a4}^{r} = F) \vee (\underline{a3}^{r} = F) \vee (\underline{a3}^{r} = F \wedge \underline{a1}^{r} = F \wedge \underline{a2}^{r} = F) \tag{3.10}$$

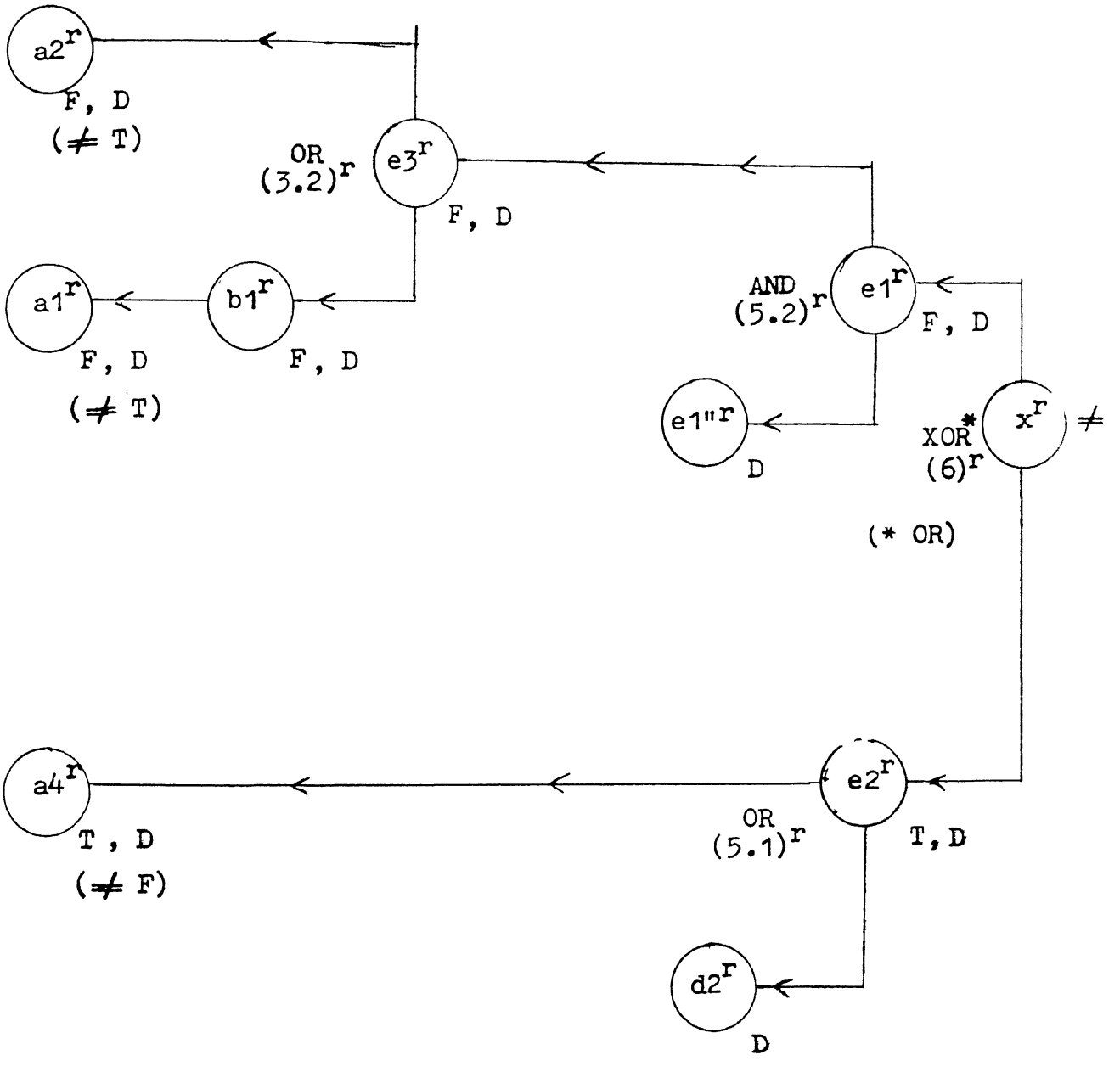


Fig.10(b): Summary of the back-tracking from a contradiction X in stage 6 for Sl. No. 4 . For the reasons given in Section 4(d), the XOR becomes OR . Also, the additional possibilities of D at the intermediate stages of the back-tracking is marked in this diagram alone.

argument from this contradiction. This can be done in exactly the same manner as in Fig. 10(a) and the summary of the results are shown in Fig. 10(b). We shall not explain this in detail except to write down the final conclusion, namely that the contradiction in stage 6 can be avoided by the conditions

$$(\underline{\underline{a2}}^r \neq F \quad \text{OR} \quad \underline{\underline{a1}}^r \neq F) \quad \text{XOR} \quad (\underline{\underline{a4}}^r \neq T) \qquad (3.11)$$

This is on the assumption that the starting point was the input T, T, F, F for $\underline{\underline{a1}}$, $\underline{\underline{a2}}$, $\underline{\underline{a3}}$, $\underline{\underline{a4}}$. In fact, the special situation of Section 4(d) occurs here, and the XOR shall be converted to OR in (3.11), and the condition becomes that either $\underline{\underline{a1}}$, or $\underline{\underline{a2}}$, or $\underline{\underline{a4}}$, or any combination of these, must be changed from the input state in Sl. No. 4, into its opposite state, (ignoring the possibility of their being D). Therefore, among the eight cases in Sl. Nos 3 to 10, only T, T, F, F in Sl. No. 4 can lead to X, taking into account also the condition that $\underline{\underline{a3}}^r$ is F from the previous retracing from stage 4. This is, in fact, verified to be the case by the detailed working out of the intermediate and final output truth values in Sl.Nos.3 to 10 of Table 5(a

(b) Reversal from a purification of D.

In Sl. Nos 3 and 4 of Table 5(a), the doubtful state produced in stage 1 gets purified in stage 4, and the effect of

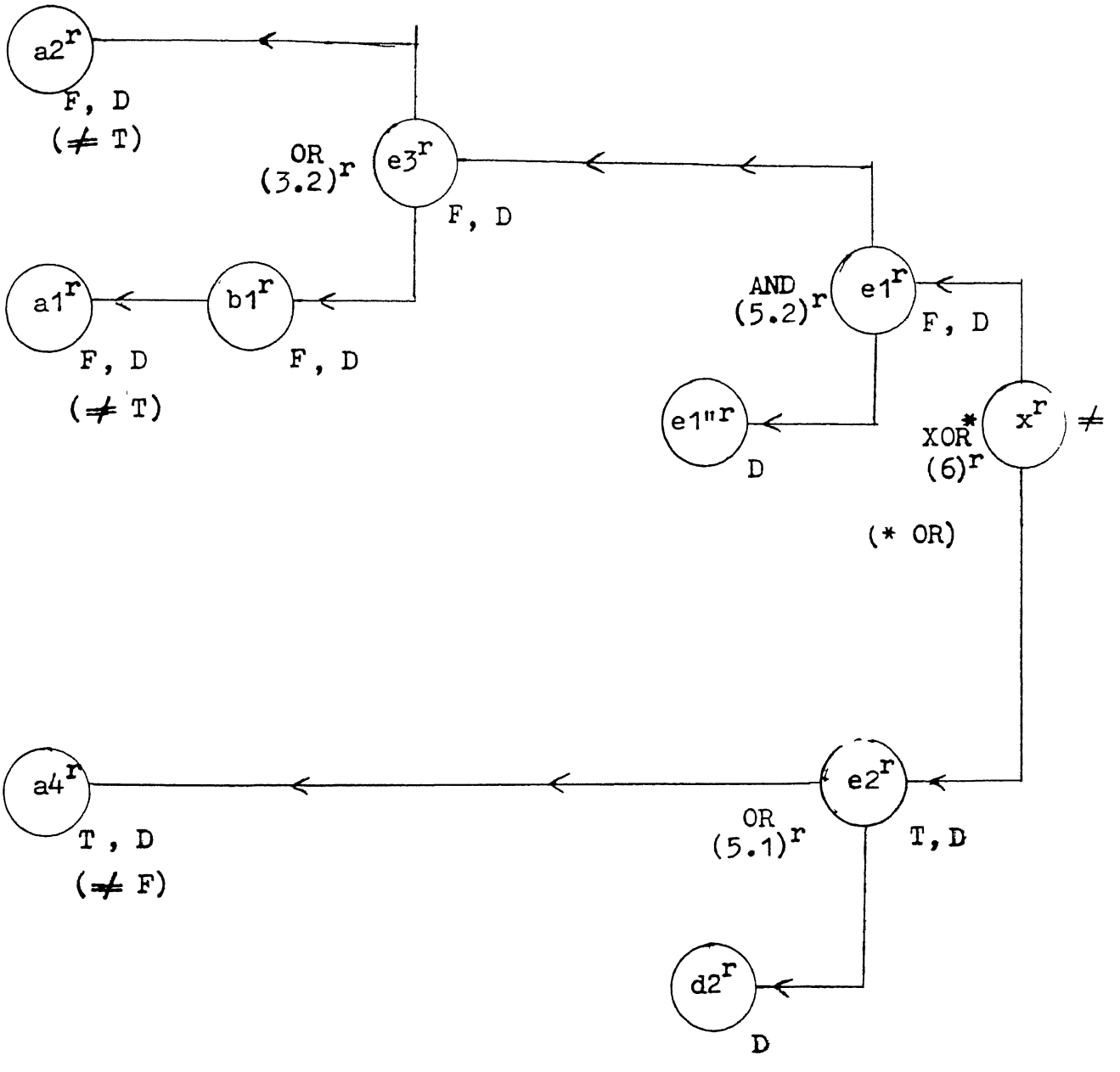


Fig.10(b): Summary of the back-tracking from a contradiction X in stage 6 for Sl. No. 4 . For the reasons given in Section 4(d), the XOR becomes OR . Also, the additional possibilities of D at the intermediate stages of the back-tracking is marked in this diagram alone.

If, however, the output is also in the D-state, then we have to accept that the solution of the graph with the given input truth values is "T or F," and the ambiguity cannot be resolved. Also, in such a case, there is no possibility of reversal from a purification stage as even the output itself is only D.

In the next lecture we shall give a brief resume of the material of this lecture in the form of rules and algorithms and then consider their extension to quantifier logic.

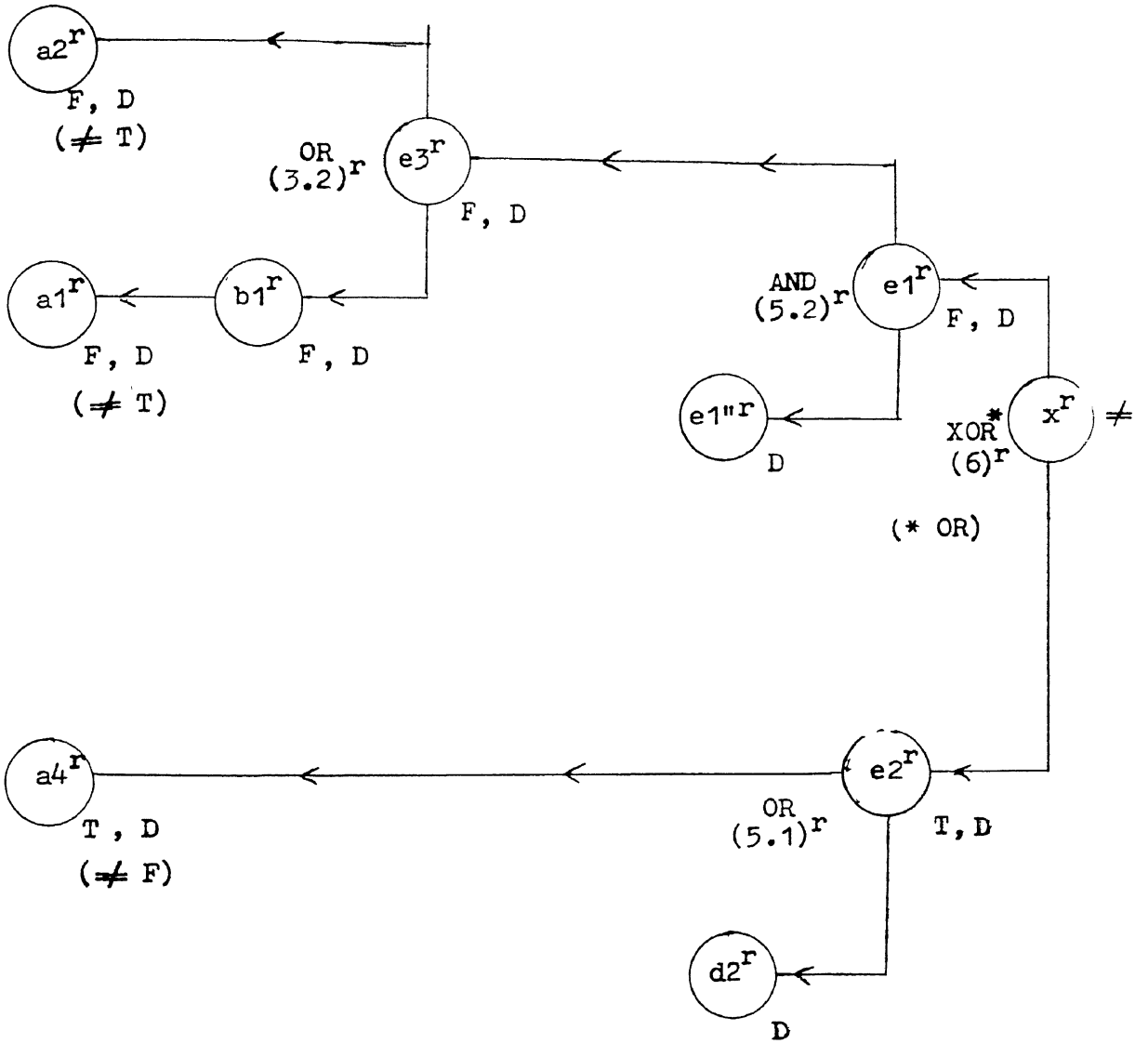


Fig.10(b): Summary of the back-tracking from a contradiction X in stage 6 for Sl. No. 4 . For the reasons given in Section 4(d), the XOR becomes OR . Also, the additional possibilities of D at the intermediate stages of the back-tracking is marked in this diagram alone.

If, however, the output is also in the D-state, then we have to accept that the solution of the graph with the given input truth values is "T or F," and the ambiguity cannot be resolved. Also, in such a case, there is no possibility of reversal from a purification stage, as even the output itself is only D.

In the next lecture we shall give a brief resume of the material of this lecture in the form of rules and algorithms and then consider their extension to quantifier logic.

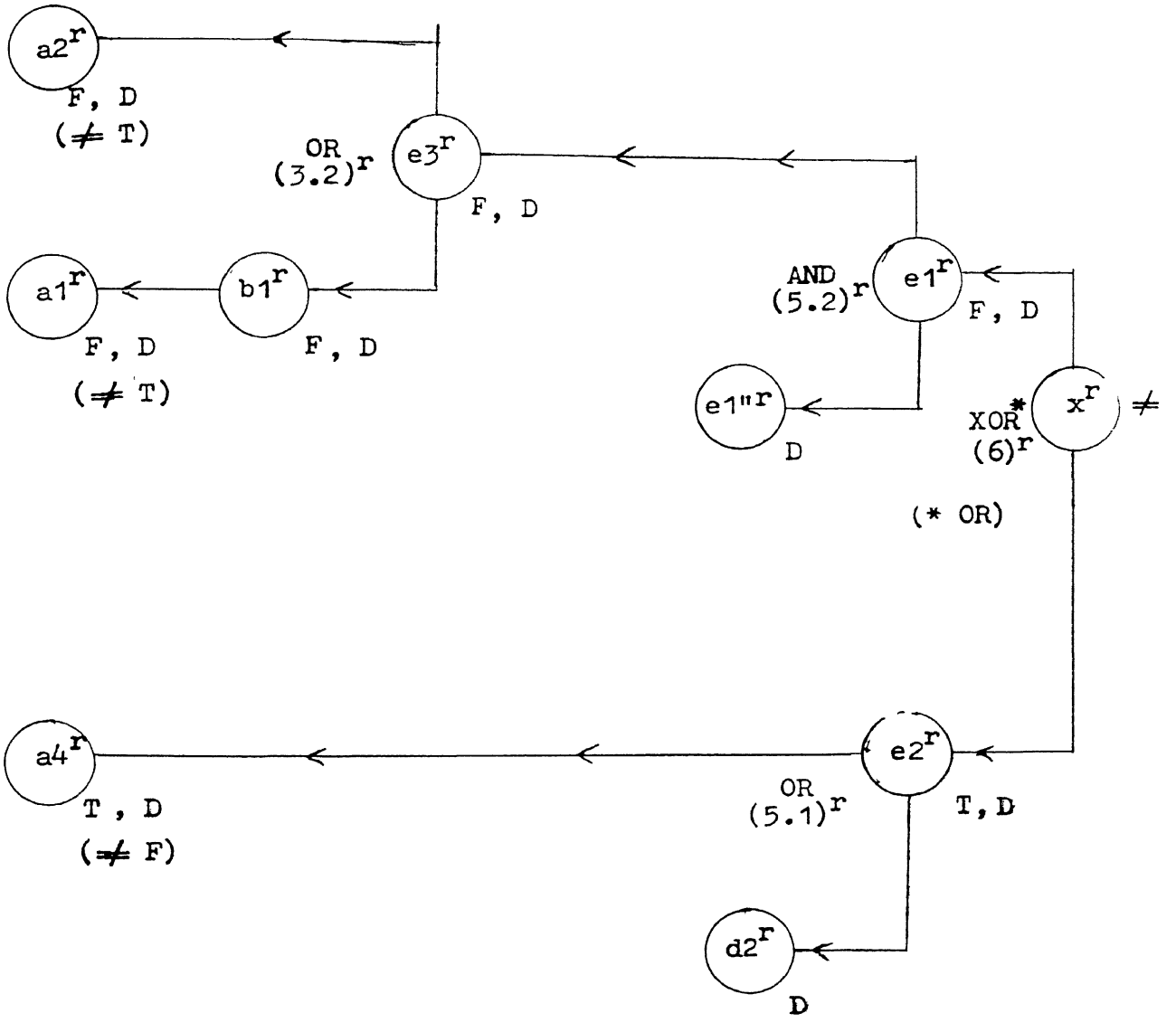


Fig.10(b): Summary of the back-tracking from a contradiction X in stage 6 for Sl. No. 4 . For the reasons given in Section 4(d), the XOR becomes OR . Also, the additional possibilities of D at the intermediate stages of the back-tracking is marked in this diagram alone.

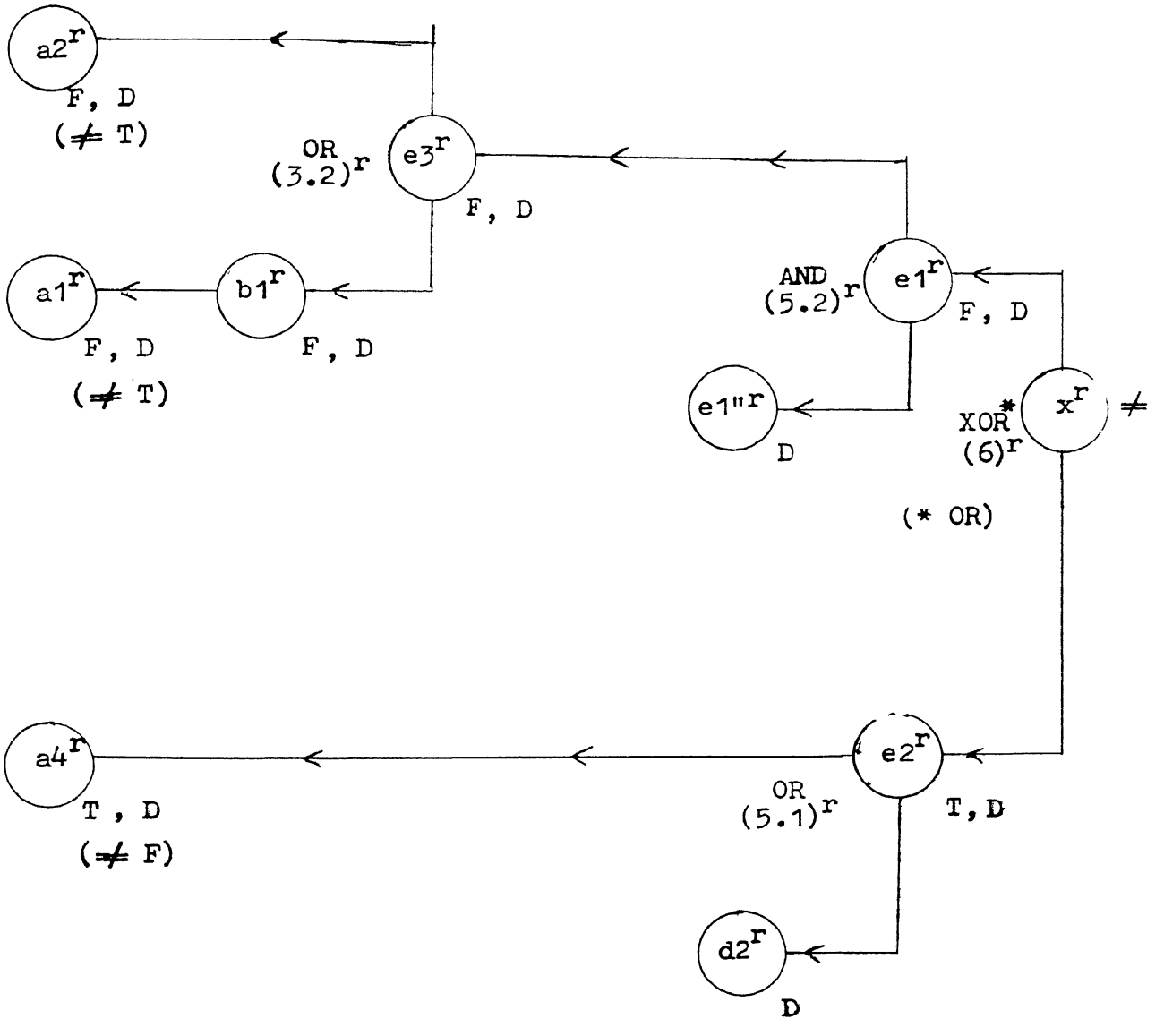


Fig.10(b): Summary of the back-tracking from a contradiction X in stage 6 for Sl. No. 4 . For the reasons given in Section 4(d), the XOR becomes OR . Also, the additional possibilities of D at the intermediate stages of the back-tracking is marked in this diagram alone.

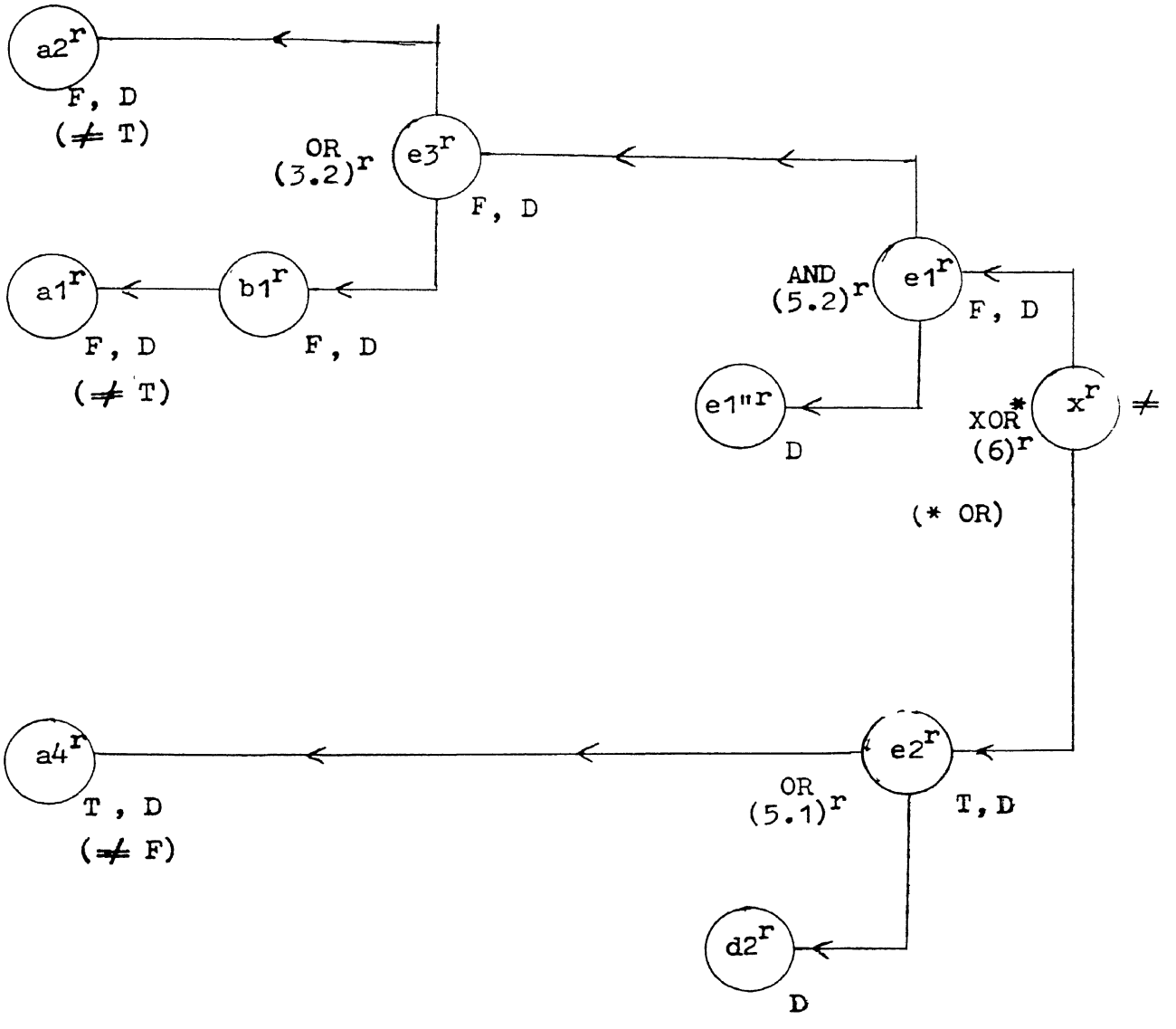


Fig.10(b): Summary of the back-tracking from a contradiction X in stage 6 for Sl. No. 4 . For the reasons given in Section 4(d), the XOR becomes OR . Also, the additional possibilities of D at the intermediate stages of the back-tracking is marked in this diagram alone.

doubtful (may be) truth values, BA and SL are both needed as pre-requisites, in addition to GA1 and GA2.

2. Classified list of subroutines in Edition 2

We give below, in Table 2, a list of all the subroutines considered here, including some which are put in for completeness although they have not been incorporated in Edition 2. We shall now comment on the footnotes to this table.

(i) Footnote * and **

Since file SL was prepared well before the files QL2 and GA1 were prepared, some of the names for subroutines were used ab initio for this purpose. Only later was it recognized that there is a complete homology between SL, QL2 and GA1, not only in theory, but also in the functions and subroutines that are required in MATLOG /, and therefore the symbology given in this table was adopted. Hence some of the subroutines like SVECEL, SVECCN, SMATEL have been given slightly different names in this edition. It will be rectified in a future edition.

(ii) Footnote ≠

In the same way, the realization that the binary product of a matrix with two vectors is more usefully denoted as a

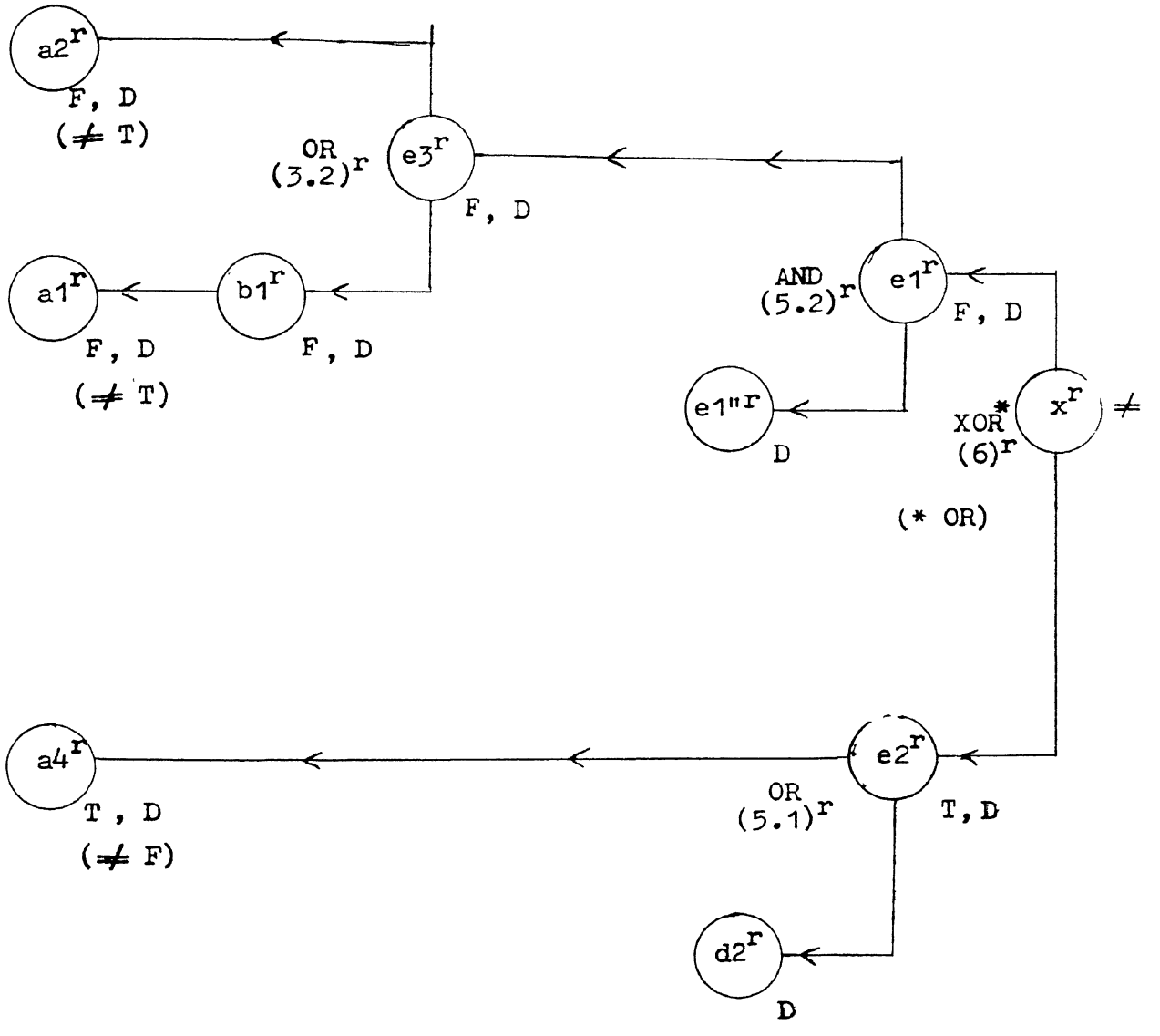


Fig.10(b): Summary of the back-tracking from a contradiction X in stage 6 for Sl. No. 4 . For the reasons given in Section 4(d), the XOR becomes OR . Also, the additional possibilities of D at the intermediate stages of the back-tracking is marked in this diagram alone.

Table 2: Contd. MR-67

Subroutines in SNS, QL-2 and GA-1

Sl. No.	File SL	File QL2	File GA1
31	SBTV2†	QBTV2†	GBTV2†
32	SSTV2	QSTV2	———
33	SUNPT2	QUNPT2	GUNPT2
34	SAGREE	QAGREE	GAGREE
35	SEQU	QEQU	GEQU
36	SNOT	QNOT	GNOT
37	SCOMP	QCOMP	GCOMP
38	SELL	QELL	GELL
51	SMXNCP	QMCON	GE
52	SMXNTR		GTC

Subroutines for QL-1 and GA-2

Sl. No.	File QL1	Sl. No.	File GA2
1	PUNION	1	GRELTV
2	PVIDYA	2	GSTV
3	PBINPT	3	GSTV2
4	PSTV		
5	PUNAND		
6	PUNOR		
7	PUNPDT		

† Not included in Edition 2

——— Included in File GA2

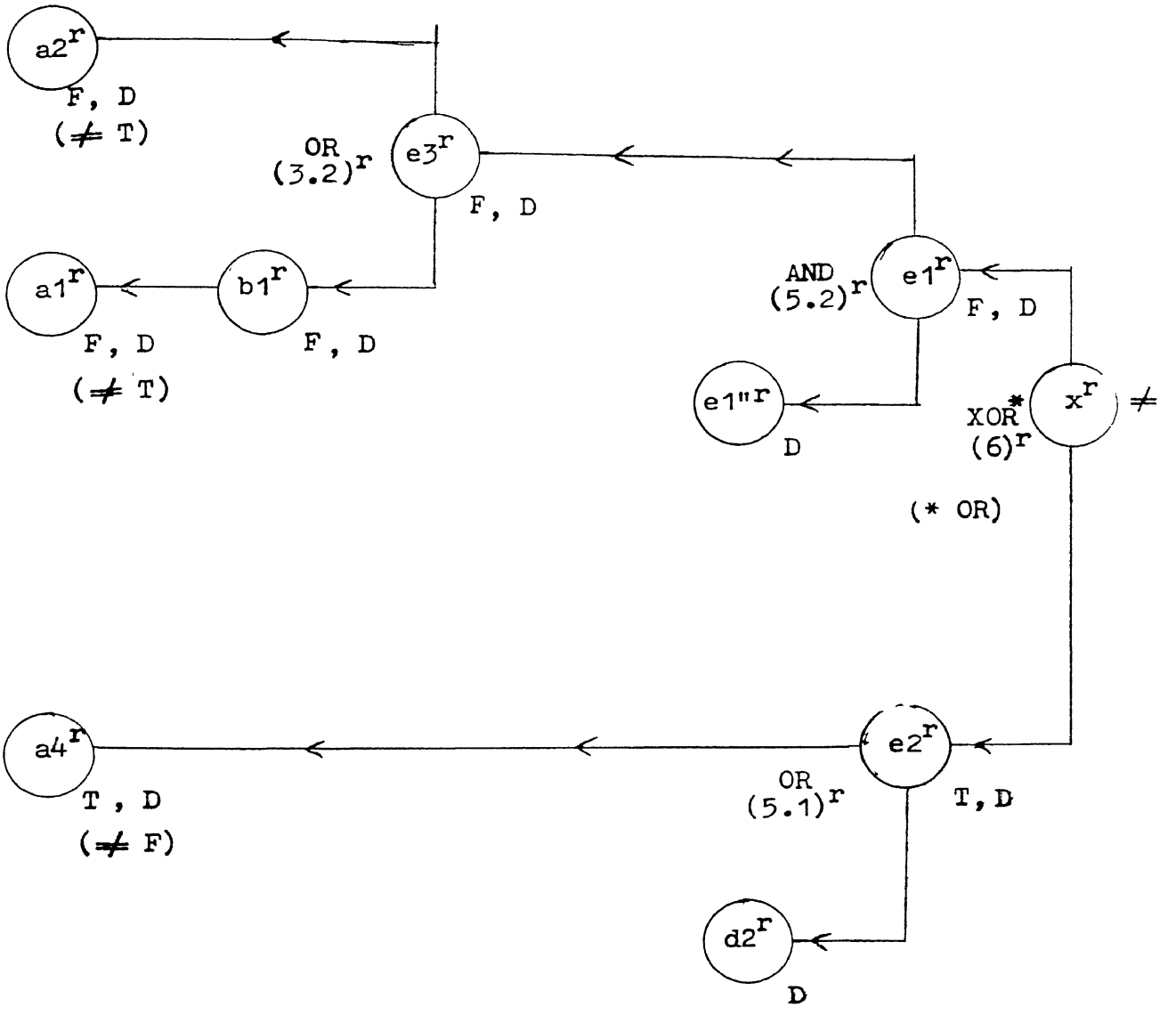


Fig.10(b): Summary of the back-tracking from a contradiction X in stage 6 for Sl. No. 4 . For the reasons given in Section 4(d), the XOR becomes OR . Also, the additional possibilities of D at the intermediate stages of the back-tracking is marked in this diagram alone.

(iv) Footnote

For file GA1, whose nature has been described above, only file BA is required as a pre-requisite and this meant that the three subroutines GA1/29, GA1/30 and GA1/32 had to be put in the file GA2 since they require SVECCN as a pre-requisite. However, Part III makes use of these subroutines as well in the theory and descriptions written there. For convenience GA1 and GA2 may be fed in, together with SL, while working out the programs for general logic. On the other hand, GA2 is intended to be a much wider file, to deal with general logic using SNS truth values quite generally. This should be written out later and the essential theory for this is contained in the Reports ALOG-31 and 32.

(v) Boolean agreement operator

One more comment may be made although it is not marked by any symbol in Table 2. This is about Sl. No. 34, namely SAGREE, QAGREE and GAGREE. The MATLOG expression for GAGREE can be very briefly written as

GAGREE = BCMP(GSCPDT(GVA,GCOMP(GVB), MI))

and it is obvious from this that its output is a Boolean scalar having the two values 1, 0 for "yes" and "no" respectively. Unnecessarily we had used the symbols T and F in SNS logic for these states. Since it saves the need for having the file SL,

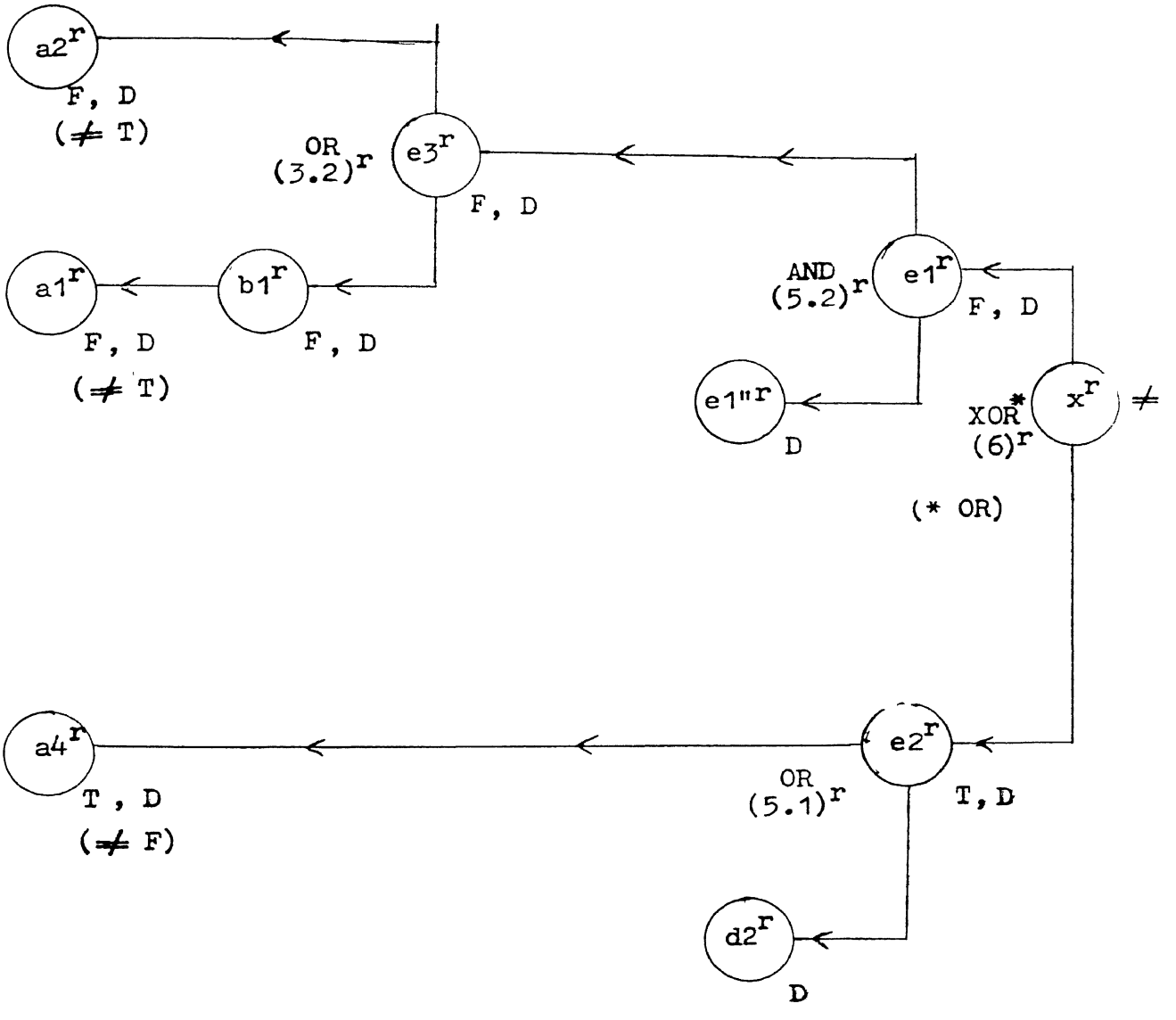


Fig.10(b): Summary of the back-tracking from a contradiction X in stage 6 for Sl. No. 4 . For the reasons given in Section 4(d), the XOR becomes OR . Also, the additional possibilities of D at the intermediate stages of the back-tracking is marked in this diagram alone.

by doing this. This is reserved for a future edition. An idea of what is envisaged in this is provided by the possible revisions of the symbols for some of the subroutines as below:

SL/32	SSTV2 (SVA, BNX, SK, SL, SVB)
SL/33	SUNPT2(SVA, BNX, SK, SL)
QL2/32	QSTV2 (QVA, BNX, QK, QL, QVB)
QL1/4	PSTV (QVZ, QVA, BNX, BK, BL, QVB, QVC, STV)
QL1/7	PUNPDT (QVZ, QVA, BNX, BK, BL)
GA1/33	GUNPT2 (GVA, BNX, GVK, GVL, MI, MJ, GVB)

It is also connected with the general possibility of using a logical equation of the type 'ANAME' = 'BNAME' in FORTRAN. Using this facility, it would be worthwnile having two subroutines BA/12 and BA/13 analogous to SL/51 and SL/52.

BA/12	BNXCP('BNX', BK, BL, 'BNXP', BKP, BLP)
BA/13	BNXTR('BNX', BK, BL, 'BNXP', BKP, BLP)

These are the only subroutines that are really needed for various purposes in QL-1 such as, for instance, for calculating the functions PSTV, PUNPDT etc.

Thus, it is quite worthwhile re-editing tnis edition after having more experience with the functions contained herein.

In particular, it should be mentioned that the PQLOGIC subroutines (QL-1B mentioned in Part II)have not been included here because these have not yet been fully debugged (for lack of technical assistance).

3. Printout of the functions and subroutines of Edition 2

The following pattern has been adopted for the functions and subroutines that are reproduced in the pages below. For each file, a list of these is given first which gives the full name of the function or subroutine, followed by its MATLOG name and the variables involved. This is followed by a brief algebraic description of the nature of the function. The input and output in each case are also clearly indicated.

The contents for each file are followed by the subroutines copied from the computer output. For convenience of reference, the serial number and the MATLOG name of each subroutine are given in the top right hand corner. Therefore, no page numbers are given for this section.

TABLE 3 : PRINTOUT OF MATLOG SUBROUTINES

FILES: BA, SL, QL2, QL1, GA1, GA2

Part A : Contents giving description of each subroutine in the various files

Part B : Printout of the subroutines in the files

List of Functions in MATLOG for basic Boolean algebra

File BA

BA/1 Boolean sum BSUM(BA, BB)

Performs Boolean addition

$$a \oplus b = c$$

Input: BA, BB Output: BSUM

BA/2 Boolean product BPDT(BA, BB)

Performs Boolean multiplication

$$a \otimes b = c$$

Input: BA, BB Output: BPDT

BA/3 Boolean equivalence BEQU(BA, BB)

Performs Boolean equivalence

$$a \Longleftrightarrow b = c$$

Input: BA, BB Output: BEQU

BA/4 Boolean complement BCMP(BA)

Performs Boolean complementation

$$a^c = b$$

Input: BA Output: BCMP

BA/5 Boolean multiple sum BMSUM(BA, MI)

Calculates Boolean multiple sum

$$a_1 \oplus a_2 \oplus \ldots \oplus a_{MI} = b$$

Input: BA(I) I = 1, MI Output: BMSUM

BA/6 Boolean multiple product BMPDT(BA, MI)

Calculates Boolean multiple product

$a_1 \otimes a_2 \otimes \ldots \otimes a_{MI} = b$

Input: BA(I), I = 1, MI Output: BMPDT

BA/7 Boolean multiple equivalence BMEQU(BA, MI)

Calculates Boolean multiple equivalence

$a_1 \Longleftrightarrow a_2 \Longleftrightarrow \ldots \Longleftrightarrow a_{MI} = b$

Input: BA(I), I = 1, MI Output: BMEQU

BA/11 Boolean truth value BTV(BA,BNX,BK,BL,BB)

Calculates Boolean truth value of a binary relation in classical logic <u>via</u> binary product of inputs a and b for the relations A(k, ℓ), O(k, ℓ), E(k, ℓ), I(k, ℓ), for (k, ℓ) = 1, 2.

e.g. $\neg a \wedge b = a\ A(2, 1)\ b = c$

Input: BA, BNX, BK, BL, BB Output: BTV

This will be included in the next edition.

List of Functions and Subroutines of MATLOG for SNS logic

File SL

SL/1 SNS vector elements SVECEL(SVA, VA)*

Finds the components a_1, a_2 of SNS vector $\underline{\underline{a}}$

$$\underline{\underline{a}} = (a_1 \quad a_2)$$

Input: SVA Output: VA(1), VA(2)

*This subroutine is named VECELE in this edition. (It will be changed later).

SL/2 SNS vector constant SVECCN(VA)*

Given the components $(a_1 \quad a_2)$ of a vector, finds the SNS constant s(k) corresponding to it.

$$(a_1 \quad a_2) = s(k)$$

Input: VA(1), VA(2) Output: VECCON

*This subroutine is named VECCON in this edition. (It will be changed later).

SL/3 SNS basic vector elements SELEM(SVA, SBAS, IB)

This will be included in the next edition.

SL/11 SNS union SUNION(SVA,SVB)

Calculates the Boolean sum of two vectors

$$\underline{\underline{a}} \oplus \underline{\underline{b}} = \underline{\underline{c}}$$

Input: SVA, SVB Output: SUNION

SL/12 SNS vidya SVIDYA(SVA, SVB)

Calculates the Boolean product of two vectors

$$\underline{\underline{a}} \otimes \underline{\underline{b}} = \underline{\underline{c}}$$

Input: SVA, SVB Output: SVIDYA

SL/13 SNS matrix sum SMXSUM(SMP, SMQ, SMR)

Calculates the Boolean sum of two matrices

$$\underline{\underline{P}} \oplus \underline{\underline{Q}} = \underline{\underline{R}}$$

Input: SMP, SMQ Output: SMR

SL/14 SNS matrix product SMXPDT(SMP, SMQ, SMR)

Calculates the Boolean product of two matrices

$$\underline{\underline{P}} \otimes \underline{\underline{Q}} = \underline{\underline{R}}$$

Input: SMP, SMQ Output: SMR

SL/15 SNS matrix complement SMXCMP(SMP, SMQ)

Calculates the Boolean complement of a matrix

$$\underline{\underline{P}}^{C} = \underline{\underline{Q}}$$

Input: SMP Output: SMQ

SL/16 SNS scalar product SSCPDT(SVA, SVB)

Calculates the scalar product of two vectors

$$\langle \underline{\underline{a}} \mid \underline{\underline{b}} \rangle = c$$

Input: SVA, SVB Output: SSCPDT

SL/17 SNS unary product SUNPDT(SVA, SMZ)

Calculates the unary product of a vector with a matrix

$$\langle \underline{\underline{a}} \mid \underline{\underline{Z}} \mid = \langle \underline{\underline{b}} \mid$$

Input: SVA, SMZ Output: SUNPDT

SL/18 SNS Boolean truth value SBTV(SVA, SMZ, SVB)

Calculates the binary product of a matrix with two vectors

$$\langle \underline{\underline{a}} \mid \underline{\underline{Z}} \mid \underline{\underline{b}} \rangle = c$$

(Calculates Boolean truth value in SNS).

Input: SVA, SMZ, SVB Output: SBINPT

*This subroutine named SBINPT in this edition. (It will be changed later).

SL/21 SNS matrix product SMATPT(SMP,SMQ,SMR)

Calculates the matrix product of two matrices

$$\bigoplus_j P_{ij} \otimes Q_{jk} = R_{ik} \quad \text{or} \quad |\underline{\underline{P}}|\underline{\underline{Q}}| = |\underline{\underline{R}}|$$

Input: SMP, SMQ Output: SMR

SL/22 SNS direct product SDIRPT(SVA, SVB, SMZ)

Performs direct product of two vectors to yield a 2x2 matrix

$$(\underline{\underline{a}} \times \underline{\underline{b}}) = \underline{\underline{Z}}$$

Input: SVA, SVB Output: SMZ

SL/23 SNS direct sum SDIRSM(SVA, SVB, SMZ)

Performs direct sum of two vectors to yield a 2x2 matrix

$$(\underline{\underline{a}} + \underline{\underline{b}}) = \underline{\underline{Z}}$$

Input: SVA, SVB Output: SMZ

SL/24 SNS direct equivalence SDIREQ(SVA, SVB, SMZ)

Performs direct equivalence of two vectors to yield a 2x2 matrix

$$(\underline{\underline{a}} \equiv \underline{\underline{b}}) = \underline{\underline{Z}}$$

Input: SVA, SVB Output: SMZ

SL/25 SNS scalar vector direct product SSVDPT(BA,SVB)

Performs scalar-vector direct product to yield a 2-vector

$$(a \times \underline{\underline{b}}) = \underline{\underline{c}}$$

Input: BA, SVB Output: SSVDPT

SL/26 SNS scalar vector direct sum SSVDSM(BA, SVB)

Performs scalar-vector direct sum to yield a 2-vector

$$(a + \underline{\underline{b}}) = \underline{\underline{c}}$$

Input: BA, SVB Output: SSVDSM

SL/27 SNS matrix elements SMATEL (SMZ, SK, SL, SM)*

Calculates the 2x2 matrices corresponding to the logical connectives

$\underline{\underline{A}}(k, \ell)$, $\underline{\underline{O}}(k, \ell)$, $\underline{\underline{E}}(k, \ell)$, $\underline{\underline{I}}(k, \ell)$

from $s(k)$ and $s(\ell)$.

Input: SMZ, SK, SL Output: SM

* **This subroutine is named MATELE in this edition.**

SL/28 SNS transpose (of matrix) STRANS(SMP, SMQ)

Calculates the transpose of an SNS matrix

$$|\underline{\underline{P}}^t| = |\underline{\underline{Q}}|$$

Input: SMP Output: SMQ

SL/29 SNS relative truth value SRELTV(SVA, SVB)

Finds the SNS relative truth value of one term, $\underline{\underline{a}}$, for another, $\underline{\underline{b}}$,

$$\underline{\underline{t}}(\underline{\underline{a}} \mid \underline{\underline{b}}) = (\langle a \mid b \rangle, \langle a \mid b^c \rangle) = (c_1, c_2)$$

Input: SVA, SVB Output: SRELTV

SL/30 SNS truth value SSTV(SVA, SMZ, SVB)

Finds the SNS truth value of the binary relation $\underline{\underline{Z}}$ for inputs $\underline{\underline{a}}$, $\underline{\underline{b}}$ (formula via contracted product)

$$\underline{\underline{t}}(\underline{\underline{a}}\ \underline{\underline{Z}}\ \underline{\underline{b}}) = (\underline{\underline{a}} \mid \underline{\underline{Z}} \mid \underline{\underline{b}}) = \underline{\underline{c}} = (c_1, c_2)$$

Input: SVA, SMZ, SVB Output: SSTV

SL/31 SNS Boolean truth value Type2 SBTV2

This will be included in the next edition

SL/32 SNS truth value Type2 SSTV2(SVA, SMZ, SK, SL, SVB)

Finds the SNS truth value of a relation involving the connective $\underline{\underline{Z}}(k, \ell)$, for inputs $\underline{\underline{a}}$, $\underline{\underline{b}}$, via relative truth values.

Input: SVA, SMZ, SK, SL, SVB Output: SSTV2

SL/33 SNS unary product Type2 SUNPT2(SVA, SMZ, SK, SL)

Finds the output of the unary relation $\underline{\underline{a}}\ \underline{\underline{Z}}(k, \ell)$ via relative truth value

$$\langle \underline{\underline{a}} \mid \underline{\underline{Z}}(k, \ell) \mid = \langle \underline{\underline{a}} \mid s(k) \rangle \; Z \; \langle s(\ell) \mid = \langle \underline{\underline{b}} \mid$$

Input: SVA, SMZ, SK, SL Output: SUNPT2

SL/34 SNS agree SAGREE(SVA, SVB)

Checks the agreement of two SNS terms and gives its Boolean truth value

$$\underline{\underline{a}}\ \underline{\underline{G}}\ \underline{\underline{b}} = (a_1 \Longleftrightarrow b_1) \otimes (a_2 \Longleftrightarrow b_2) = c$$

Input: SVA, SVB Output: SAGREE

SL/35 SNS equivalence (of vector) SEQU(SVA)

Performs the matrix-Boolean unary operation $\underline{\underline{E}}$ to yield

$$\underline{\underline{a}}\ \underline{\underline{E}} = \underline{\underline{b}}$$

Input: SVA Output: SEQU

SL/36 SNS negation (of vector) SNOT(SVA)

Performs the matrix-boolean unary operation $\underline{\underline{N}}$ to yield

$$\underline{\underline{a}}\ \underline{\underline{N}} = \underline{\underline{b}}$$

Input: SVA Output: SNOT

SL/37 SNS complement (of vector) SCOMP(SVA)

Performs matrix-boolean unary operation $\underline{\underline{M}}$ to yield

$$\underline{\underline{a}}\ \underline{\underline{M}} = \underline{\underline{b}}$$

Input: SVA Output: SCOMP

SL/38 SNS ellation (of vector) SELL(SVA)

Performs the matrix-boolean operation $\underline{\underline{L}}$ to yield

$$\underline{\underline{a}}\ \underline{\underline{L}} = \underline{\underline{b}}$$

Input: SVA Output: SELL

SL/51 SNS matrix name complement SMXNCP(SMX,SK,SL,SMXP, SKP, SLP)

Performs name-complement of SNS connective $\underline{\underline{Z}}(k, \ell)$ to yield

$$\underline{\underline{Z}}^{c}(k,\ell) = \underline{\underline{Z}}'(k', \ell')$$

Input: SMX, SK, SL Output: SMXP, SKP, SLP

SL/52 SNS matrix name transpose SMXNTR(SMX,SK,SL,SMXP,SKP,SLP)

Performs name-transpose of SNS connective $\underline{\underline{Z}}(k, \ell)$ to yield

$$\underline{\underline{Z}}^{t}(k,\ell) = \underline{\underline{Z}}'(k', \ell')$$

Input: SMX, SK, SL Output: SMXP, SKP, SLP

List of Functions and Subroutines of MATLOG for Quantifier Logic

Type2

File QL2

QL2/1 QLOGIC vector elements QVECEL(QVA, VA)

Performs decomposition of 3-vector into its components

$\underset{\sim}{a} = (a_1\ a_2\ a_3) \longmapsto q(1), q(3), q(5)$, (those present in $\underset{\sim}{a}$)

Input: QVA Output: VA(1), VA(2), VA(3).

QL2/2 QLOGIC vector constant QVECCN(VA)

Given the components of vector, finds the state

$(a_1\ a_2\ a_3) \longmapsto q(\underset{\sim}{k}),\ \underset{\sim}{k} = 1, 8$

Input: VA(1), VA(2), VA(3) Output: QVECCN

QL2/3 Quantifier basic vector elements QELEM(QVA,QBAS,IB)

Decomposes a vector into its basic vectors

Input: QVA Output: QBAS, IB

QL2/11 QLOGIC union QUNION(QVA, QVB)

Calculates the Boolean sum of two vectors

$$\underset{\sim}{a} \oplus \underset{\sim}{b} = \underset{\sim}{c}$$

Input: QVA, QVB Output: QUNION

QL2/12 QLOGIC vidya QVIDYA(QVA, QVB)

Calculates the Boolean product of two vectors

$$\underset{\sim}{a} \otimes \underset{\sim}{b} = \underset{\sim}{c}$$

Input: QVA, QVB Output: QVIDYA

QL2/13 QLOGIC matrix sum QMXSUM(QMP,QMQ,QMR)

Calculates the Boolean sum of two matrices

$$\underset{\sim}{P} \oplus \underset{\sim}{Q} = \underset{\sim}{R}$$

Input: QMP, QMQ Output: QMR

QL2/14 QLOGIC matrix product QMXPDT(QMP,QMQ,QMR)

Calculates the Boolean product of two matrices

$$\underset{\sim}{P} \otimes \underset{\sim}{Q} = \underset{\sim}{R}$$

Input: QMP, QMQ Output: QMR

QL2/15 QLOGIC matrix complement QMXCMP(QMP, QMQ)

Calculates the Boolean complement of a matrix

$$\underset{\sim}{P}^{c} = \underset{\sim}{Q}$$

Input: QMP Output: QMQ

QL2/26 QLOGIC scalar vector direct sum QSVDSM(BA,QVB)

Performs scalar-vector direct sum

$$(a + \underset{\sim}{b}) = \underset{\sim}{c}$$

Input: BA, QVB Output: QSVDSM

QL2/27 QLOGIC matric elements QMATEL(QMZ,QK,QL,QM)

Finds the matrix elements of logical connectives

$\underset{\sim}{A}(\underset{\sim}{k}, \underset{\sim}{\ell})$, $\underset{\sim}{O}(\underset{\sim}{k}, \underset{\sim}{\ell})$, $\underset{\sim}{E}(\underset{\sim}{k}, \underset{\sim}{\ell})$, $\underset{\sim}{I}(\underset{\sim}{k}, \underset{\sim}{\ell})$

Input: QMZ, QK, QL Output: QM

QL2/28 QLOGIC transpose (of matrix) QTRANS(QMP, QMQ)

Calculates the transpose of a matrix

$$|\underset{\sim}{P}^t| = |\underset{\sim}{Q}|$$

Input: QMP Output: QMQ

QL2/29 QLOGIC relative truth value QRELTV(QVA,QVB)

Finds the SNS relative truth value of one quantified term $\underset{\sim}{a}$, for another $\underset{\sim}{b}$,

$$\underline{\underline{t}}(\underset{\sim}{a} | \underset{\sim}{b}) = \underline{\underline{c}}$$

Input: QVA, QVB Output: QRELTV

QL2/30 QLOGIC truth value QSTV(QVA, QMZ, QVB)

Finds the SNS truth value of a binary relation for given inputs $\underset{\sim}{a}$, $\underset{\sim}{b}$ and the matrix elements of the connective $\underset{\sim}{Z}$.

$$\underline{\underline{t}}(\underset{\sim}{a}\ \underset{\sim}{Z}\ \underset{\sim}{b}) = (\underset{\sim}{a}|\underset{\sim}{Z}|\underset{\sim}{b}) = (c_1\ \ c_2) = \underline{\underline{c}}$$

Input: QVA, QMZ, QVB Output: QSTV

QL2/31 QLOGIC Boolean truth value Type2 QBTV2

This will be included in the next edition.

QL2/32 QLOGIC truth value Type2 QSTV2(QVA,QMZ,QK,QL,QVB)

Finds the SNS truth value of a relation involving the QL-2 logical connective $\underset{\sim}{Z}(\underset{\sim}{k}, \underset{\sim}{\ell})$ via relative truth values.

Input: QVA,QMZ,QK,QL,QVB Output: QSTV2

QL2/33 QLOGIC unary product Type2 QUNPT2(QVA,QMZ,QK,QL)

Finds the output of the unary relation $\underset{\sim}{a}\ \underset{\sim}{Z}(\underset{\sim}{k}, \underset{\sim}{\ell})$ via relative truth value

$$\langle \underset{\sim}{a} \,|\, \underset{\sim}{Z}(\underset{\sim}{k}, \underset{\sim}{\ell}) \,| = \langle \underset{\sim}{a} \,|\, q(\underset{\sim}{k}) \rangle \; Z \; \langle q(\underset{\sim}{\ell}) \,| = \langle \underset{\sim}{b} \,|$$

Input: QVA, QMZ, QK, QL Output: QUNPT2

QL2/34 QLOGIC agree QAGREE(QVA,QVB)

Checks the agreement of the vector components of two quantified terms and gives its Boolean truth value

$$\underset{\sim}{a}\ \underset{\sim}{G}\ \underset{\sim}{b} = (a_1 \Longleftrightarrow b_1) \otimes (a_2 \Longleftrightarrow b_2) \otimes (a_3 \Longleftrightarrow b_3) = c$$

Input: QVA, QVB Output: QAGREE

QL2/35 QLOGIC equivalence (of vector) QEQU(QVA)

Performs matrix-boolean unary operation $\underset{\sim}{E}$ to yield

$$\underset{\sim}{a}\ \underset{\sim}{E} = \underset{\sim}{b}$$

Input: QVA Output: QEQU

QL2/36 QLOGIC negation (of vector) QNOT(QVA)

Performs matrix-boolean unary operation $\underset{\sim}{N}$ to yield

$$\underset{\sim}{a}\,\underset{\sim}{N} = \underset{\sim}{b}$$

Input: QVA Output: QNOT

QL2/37 QLOGIC complement (of vector) QCOMP(QVA)

Performs matrix-boolean unary operation $\underset{\sim}{M}$ to yield

$$\underset{\sim}{a}\,\underset{\sim}{M} = \underset{\sim}{b}$$

Input: QVA Output: QCOMP

QL2/38 QLOGIC ellation (of vector) QELL(QVA)

Performs matrix-boolean unary operation $\underset{\sim}{L}$ to yield

$$\underset{\sim}{a}\,\underset{\sim}{L} = \underset{\sim}{b}$$

Input: QVA Output: QELL

QL2/51 QLOGIC matrix connective QMCON(QVK,SMX,SK,SL, QVL,QMZ,QK,QL)

Finds matrix connective in QLOGIC via matrix connective in SLOGIC

$$\underset{\sim}{q}(\underset{\sim}{k}_Z)\ \ \underset{\equiv}{Z}(k,\ell)\ \ \underset{\sim}{q}(\underset{\sim}{\ell}_Z) = \underset{\sim}{Z}(\underset{\sim}{k},\ \underset{\sim}{\ell})$$

Input: QVK, SMX,SK,SL,QVL Output: QMZ,QK,QL

List of Functions and Subroutines of MATLOG for quantifier logic

Type1

File QL1

QL1/1 PLOGIC union PUNION(QVA,QVB)

Performs QL-1 binary 'OR'

$$\underset{\sim}{a}x \; \underline{\underline{O}} \; \underset{\sim}{b}x = \underset{\sim}{c}x$$

Input: QVA, QVB Output: PUNION

QL1/2 PLOGIC vidya PVIDYA(QVA,QVB)

Performs QL-1 binary "AND"

$$\underset{\sim}{a}x \; \underline{\underline{A}} \; \underset{\sim}{b}x = \underset{\sim}{c}x$$

Input: QVA, QVB Output: PVIDYA

QL1/3 PLOGIC binary product PBINPT(QVA,SMX,SK,SL,QVB)

Calculates the QL-1 binary product of quantified terms $\underset{\sim}{a}$, $\underset{\sim}{b}$ with the SNS operator $\underline{\underline{Z}}(k, \ell)$

$$\underset{\sim}{a}x \; \underline{\underline{Z}}(k, \ell) \; \underset{\sim}{b}x = \underset{\sim}{c}x$$

Input: QVA,SMX,SK,SL,QVB Output: PBINPT

QL1/4 PLOGIC SNS truth value PSTV(QVZ,QVA,SMX,SK,SL,QVB, QVC, STV)

Calculates the binary truth value of a standard QL-1A relation for given inputs, namely

$\underset{\sim}{a}x$, $\underset{\sim}{b}x$, $(\underset{\sim}{q}_Z x)$ $(\underset{\sim}{a}x \; \underline{\underline{Z}}(k, \ell) \; \underset{\sim}{b}x) = \underline{\underline{t}}$ as

$\underset{\sim}{a}x \; \underline{\underline{Z}}(k, \ell) \; \underset{\sim}{b}x = \underset{\sim}{c}$, and $\langle \underset{\sim}{q}_Z x \mid \underset{\sim}{c} \rangle = \underline{\underline{t}}$

Input: QVZ,QVA,SMX,SK,SL,QVB Output: QVC, PSTV

QL1/5 PLOGIC unary 'AND' PUNAND(QVZ,QVA)

Performs unary 'AND' operation in QL-1A

$$\underset{\sim}{a}x,\ (q_{\underset{\sim}{Z}}x)\ (\underset{=}{a}x\ \underline{A}\ \underset{=}{b}x) \longmapsto \underset{\sim}{b}x$$

Input: QVZ, QVA Output: PUNAND

QL1/6 PLOGIC unary 'OR' PUNOR(QVZ, QVA)

Performs unary 'OR' operation in QL-1A

$$\underset{\sim}{a}x,\ (q_{\underset{\sim}{Z}}x)\ (\underset{=}{a}x\ \underline{O}\ \underset{=}{b}x) \longmapsto \underset{\sim}{b}x$$

Input: QVZ, QVA Output: PUNOR

QL1/7 PLOGIC unary product PUNPDT(QVZ,QVA,SMX,SK,SL)

Calculates the unary product of a standard QL-1A relation

$$\underset{\sim}{a}x,\ (q_{\underset{\sim}{Z}}x)\ (\underset{=}{a}x\ \underline{Z}(k,\ell)\ \underset{=}{b}x) \longmapsto \underset{\sim}{b}x$$

Input: QVZ,QVA,SMX,SK,SL Output: PUNPDT

List of Functions and Subroutines of MATLOG for GLOGIC

Theory of relations with Boolean truth values

File GA1

GA1/1 GLOGIC vector elements GVECEL(GVA,VA)

This will be included in the next edition.

GA1/2 GLOGIC vector constant GVECCN(VA)

This will be included in the next edition.

GA1/3 GLOGIC basic vector elements GELEM(GVA,GBAS,IB)

This will be included in the next edition.